Responsiv webbutveckling med Joomla

Pyry Strandén

Examensarbete

Informations- och medieteknik

2013

Pyry Strandén

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	4185
Författare:	Pyry Strandén
Arbetets namn:	Responsiv webbutveckling med Joomla
Handledare (Arcada):	Johnny Biström
Uppdragsgivare:	

Sammandrag:

Responsiv design är ett begrepp som Ethan Marcotte gjorde känd i sin artikel ”Responsive Web Design” i webbtidningen ”A list apart” år 2010. I dagens läge vet en webbutvecklare sällan i förhand med hurdan apparat människorna besöker sina publikationer med. Det kan vara en smarttelefon, surfplatta, bordsdator eller t.o.m. en television. Alla dessa apparater kan ha olika skärmstorlekar och funktioner och ställer då också krav på hur en webbutvecklare planerar sin webbplats. Att planera en webbplats skilt för alla apparater skulle vara både tidskrävande och ofta ekonomiskt olönsamt. Dessa problem ville Marcotte lösa i sin artikel och senare i boken ”Responsive Web Design” 2011. Han kombinerade tre existerande metoder till en gemensam och kallade den för responsiv design. Detta sätt att bygga sidor började snabbt slå igenom bland webbutvecklarna. En responsiv webbsida är mera än en sida som bara anpassar sig till den apparat och skärmstorlek som slutanvändaren besöker sidorna med. Detta slutarbete kommer att koncentrera sig på hur man utvecklar en responsiv webbplats i Joomla innehållshanteringssystemet. Att använda ett innehållshanteringssystem har blivit ett allt vanligare sätt att publicera innehåll i internet och målet med arbetet är att bygga en responsiv sida i Joomla genom att använda de nyaste CSS3 tekniker kombinerat med Twitter Bootstrap ramverket som finns färdigt implementerat i Joomla. Arbetet går igenom de viktigaste elementen för en responsiv design och hur man använder dessa element i Joomla miljön.

Nyckelord:	Responsiv webbutveckling, Joomla, Twitter Bootstrap, Media Query, flexibel grid, CSS3, CMS, HTML5
Sidantal:	54
Språk:	Svenska
Datum för godkännande:	24.4.2013

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media Technology
Identification number:	4185
Author:	Pyry Strandén
Title:	Responsiv webbutveckling med Joomla
Supervisor (Arcada):	Johnny Biström
Commissioned by:	

Abstract:

Responsive design is a term that Ethan Marcotte made known in his article “Responsive Web Design” in the webzine “A list apart” year 2010. Nowadays a web developer seldom knows beforehand with what kind of device the user is visiting his publications. It can be a smartphone, tablet, desktop computer or even a TV. All of these devices can have different screen sizes and functionality and because of that set requirements on how the web developer designs his pages. To make a separate design for all of these devices would be time-consuming and often economically unwise. These were the problems that Marcotte wanted to solve in his article and later in the book “Responsive Web Design” 2011. He combined a few existing techniques into one approach and called it responsive design and that way of thinking quickly started to gain popularity in the web developing society. A responsive webpage is more than just a page that adapts according to the device and screen resolution that the user is using browsing the page with. This thesis concentrates on the development of responsive web pages to the Joomla content management system. The use of CMS for publishing content in the Internet has become more common and the purpose of this project is to build a responsive website using the newest CSS3 techniques combined with the Twitter Bootstrap framework that comes preinstalled in Joomla. The thesis will go through the most important elements of responsive web development and how to implement them to the Joomla working environment.

Keywords:	Responsive web development, Joomla, Twitter Bootstrap, Media Query, flexible grid, CSS3, CMS, HTML5
Number of pages:	54
Language:	Swedish
Date of acceptance:	24.4.2013

INNEHÅLL

FIGURER

Tabeller

1 INLEDNING

Användande av Internet utanför hemmet och jobbet har blivit allt vanligare. Vi använder oss allt mera av mobila apparater som smarttelefoner och surfplattor. Försäljningen av dessa har ökat drastiskt under de senaste åren och vi surfar på nätet var vi än befinner oss. Liggande på soffan eller under arbetsresan har vi nästan alltid tillgång till Internet med en mobilapparat. Detta ställer krav på hur en webbsida bör se ut och fungera. Tidigare formade man en webbsida med en bestämd bredd t.ex. 960 pixlar som såg relativt bra ut för en laptop, men kanske för liten för en stationär dator med en stor skärm. Men idag kan en webbsida inte se ut och fungera på samma sätt på en 4 tums smarttelefon, en 10 tums surfplatta och en 22 tums datorskärm. En lösning som man använder sig av, är att göra skilda sidor för mobila apparater vilka man kan kännetecknа med ett m- eller mobi underdomän t.ex. *mobi.iltasanomat.fi*. Men för att designa och utföra flera olika nätsidor för olika apparater skulle fordra mera arbetstimmar och större underhållskostnader. Svaret på dessa krav är responsiva webbsidor, vilka tillämpar sig för den apparat som slutanvändaren besöker sidorna med. Konceptet är inte nytt och tekniken bakom det har funnits länge, men först under de senaste åren har man på allvar börjat dra nytta av denna teknik.

För att underlätta arbetet inom webbutvecklingen finns det olika CMS, content managment system, eller på svenska innehållshanteringssystem som baserar sig på en öppen källkod eller också betalda versioner. De innehåller oftast ett grafiskt användargränssnitt som t.ex. kan bestå av ordbehandlingsprogram eller bildbehandlingsprogram med vilka det är lättare för användaren att själv uppdatera sidorna. Detta möjliggör för dem, som inte är så bekanta med att koda eller bygga webbsidor, att publicera eller underhålla bloggar eller en fullständig webbplats.

Med den ökande användnigen av mobila apparater måste innehållshanteringsystemen också vara kapabla att publicera responsiva webbsidor. De tre populäraste är Drupal, Joomla och Wordpress. Med publiceringen av den senaste versionen av Joomla i september år 2012, har Joomla koncentrerat sig mera på responsiva egenskaper och webutveckling till mobila apparater. Detta är också den viktigaste orsaken till varför jag valde

att forska i detta ämne. Joomla sägs ha blivit lite på efterkälken bland de populära CMS, men med Joomla 3 hoppas de igen bli en alvarlig kandidat i CMS marknaden.
(Joomla!, 2013)

1.1 Syfte

Med detta arbete vill jag undersöka hur man producerar en responsiv webbsida med Joomla. Joomla är den näst mest använda CMS med 8,5 % av alla CMS-användare i jämförelse med Wordpress 54,8 % och Drupal 7,1 % (W3techs, 2013). Jag kommer att gå igenom de största kraven på en responsiv sida och vad man bör tänka på när man planerar en sida som kommer att ha mobila besökare. Jag kommer att undersöka hur man producerar en responsiv sida som tillfredsställer alla dagens krav, vilka egenskaper en responsiv sida i Joomla kan ha och om det är tillräckligt flexibelt att bygga en responsiv sida som grundar sig på Joomla samt hur enkelt sidans planering är. Ett problem med att använda innehållshanteringssystem är de långa laddningstiderna då man använder en mobilapparat. Jag kommer att undersöka om man med några enkla optimeringstekniker kan förbättra på detta. Den slutliga produkten bör vara stödd av alla moderna webbläsare och användningen av sidorna med en liten skärm ska vara mera ett nöje än en plåga. Då sidorna är färdiga kommer jag att undersöka hur bra de fungerar på de olika plattformarna och hur lätt det är för en slutanvändare att göra små förändringar i sidorna.

1.2 Avgränsning

Arbetet kommer att avgränsas till att undersöka Joomlas egenskaper i den responsiva utvecklingen samt hur ramverket Bootstrap används i skapandeprocessen. Arbetet kommer att kort gå igenom vad en responsiv sida är och vad det krävs för att bygga en responsiv sida, men själva skapandeprocessen kommer att ske i Joomla miljön och inga andra CMS-system kommer att forskas eller jämföras. Jag kommer inte att gå igenom hur Joomla installeras eller vad som krävs för att köra Joomla, inte heller hur Joomlas databaser och dynamiska sidor fungerar jämfört med en statisk webbsida. Joomlas basfunktioner, som skapandet av artiklar och annat innehåll till sidorna, kommer bara att gås igenom från en responsiv synvinkel, d.v.s. hur bra allt detta fungerar med en smart-

telefon eller surfplatta jämfört med en persondator. Ingen färdig webbplats kommer heller att byggas, utan bara en mall av en webbsida för att demonstrera hur man bygger den och vilka egenskaper man kan involvera i en responsiv Joomla webbsida.

1.3 Metoder

Jag kommer att göra litteraturstudier om responsiv utveckling för att bekanta mig med de senaste teknikerna. Jag studerar manualer och teknisk dokumentation som kommer från Joomla samt rekommendationer från W3C och jämför hur bra de stämmer överens med varandra. För att bygga sidan kommer experiment att göras för att se hur man får fram det bästa resultatet. Den slutliga webbsidan kommer jag att testa med följande apparater: Samsung Galaxy S, Apple iPhone 4s, Asus Transformer TF300t och normala bordsdatorer.

1.4 Terminologi och förkortningar

CSS3 – Cascading Style Sheets. Stilmall som beskriver presentationsstilen för en sida. CSS3 innehåller nya tekniker vilket underlättar skapandet av responsiva sidor

CMS – Content Managment System eller innehållshanteringssystem på svenska. Ett program som underlättar hanteringen av innehåll. Innehållet finns lagrat i databaser vilka användaren kan lätt plocka fram med ett användarvänligt sätt.

Responsiv – En responsiv webbsida innebär att innehållet tillämpar sig till den skärmupplösning eller apparat besökaren har, vilket underlättar användningen av sidan med t.ex. en smarttelefon.

Open Source – Öppen källkod. Oftast datorprogram där källkoden är öppen för användaren att öppna, läsa, skriva och vidaredistribuera.

Joomla – Ett innehållshanteringssystem som baserar sig på öppen källkod och möjliggör en lättare produktion och hantering av webbplatser.

GitHub – En server för serverhantering. Gått förbi Sourceforge och Google Code som den populäraste webbplatsen för mjukvaruutveckling. Finns på github.com. (Read and Write, 2013)

Grid – Ett rutsystem där man med hjälp av horisontella och vertikala guidelinjer placerar innehåll på en eller flera sidor för en konsekvent och visuellt attraktiv layout

Flexibel/fixed grid – En flexibel grid anpassar sig till användarens skärmstorlek medan en fixed grid alltid är lika stor.

Smarttelefon – En mobil enhet som kan fungera som en mobiltelefon så väl som att den har vissa samma funktioner som en stationär hemdator.

WAMP – Ett färdigt mjukvarupaket för Windows maskiner som innehåller tre individuellt skapade komponenter: Apache, MySQL och ett av PHP, Perl eller Python. Apache är en webbserver, MySQL är en databas och PHP, Perl och Python är skriptspråk som kan behandla information från databaser och skapa dynamiska sidor då webbläsaren gör förfrågningar för information. WAMP är då en akronym av Windows, Apache, MySQL, och PHP/Perl/Python.

Media Query – Ett logiskt uttryck som är antingen sann eller falsk. Med detta uttryck kan man göra förfrågningar till webbläsaren, vilket möjliggör en ändring i innehållet för att anpassa sig t.ex. med skärmstorleken.

Div – HTML element som fungerar som en allmän behållare för innehåll. Divarna kan innehålla text, bilder, navigation eller annat innehåll.

Template – En mall i Joomla som används som botten för Joomlas webbsidor. Templates är det som webbutvecklare blir tvungna att bygga för att få den funktionalitet och det utseende han vill ha på sin Joomla webbsida.

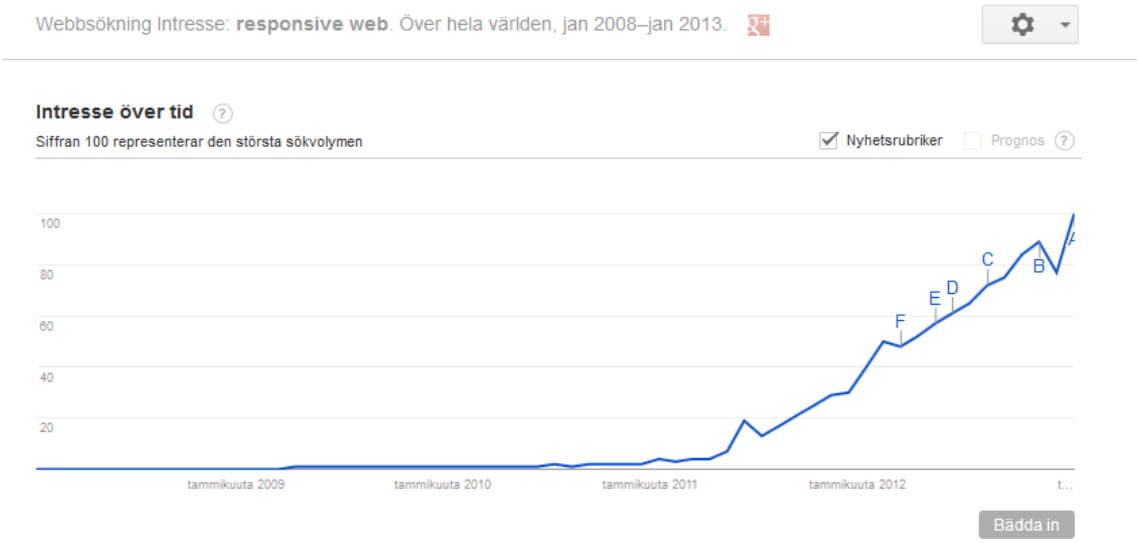
2 RESPONSIVA WEBBEN

Med den allt ökande användningen av smarttelefoner och surfplattor har responsiva webbsidor, d.v.s. webbsidor som tillämpar sig för den skärmstorleken som används, blivit ett allt viktigare ämne inom webbutvecklingen. Det krävs inget nytt programmeringsspråk eller nya verktyg, utan man kan koncentrera sig på att använda de verktyg som redan finns för att planera bättre och mera användbara sidor oberoende av vilken skärmstorlek eller apparat man använder. För några år sedan såg alla sidor exakt likadana ut på en mobiltelefon med liten skärm och en bordsdator med stor skärm.

Figur 1. ITV.com på en smarttelefon för några år sedan och en responsiv version i dagens läge (Frain, 2012)

I figur 1 kan man se hur en sida kunde se ut på en smarttelefon för några år sedan och hur samma sida idag ser ut, då den anpassar sig till användarens skärmstorlek. I den första versionen blev man tvungen att hela tiden zooma in och flytta på skärmen åt höger och vänster för att kunna läsa en text som ursprungligen var gjord för större skärmar. På högra sidan finns en responsiv sida där man inte behöver zooma in och ut, utan sidan är planerad att vara användbar såväl till smarttelefoner, surfplattor och större skärmar.

En sökning i Google Trends visar att fastän konceptet inte är nytt, har behovet och intresset för responsiv utveckling ökat drastiskt efter mitten av 2011.



Figur 2. Responsive web på Google Trends. Bildkapning (Google Trends, 2013)

Behovet kommer inte att minska eftersom det, enligt Canalys, redan år 2011 såldes 488 miljoner smarttelefoner mot 418 miljoner persondatorer. Trenden kommer att fortsätta då det t.ex. i utvecklingsländer är billigare att äga en 3G och 4G koppling än en bredbandskoppling och människorna där kanske inte har möjligheter att skaffa en dyr persondator och nöjer sig med en billigare lösning i form av en surfplatta eller smarttelefon. I februari 2010 var 1,7 % av alla sidvisningar i Internet gjorda med en mobilapparat och i februari 2013 hade sidvisningar för mobilapparater stigit till 14,4 %. År 2016 uppskattas det enligt Canalys att 2,6 miljarder mobila apparater kommer att säljas. (ITWire 2012, Statscounter, 2013, Canalys, 2012)

Worldwide mobile device shipments

Category	2012 shipments (millions)	2016 shipments (millions)	CAGR
Total	1,936.2	2,614.2	7.8%
Basic phone	122.0	58.0	-17.0%
Feature phone	770.8	660.9	-3.8%
Smart phone	694.8	1,342.5	17.9%
Tablet	114.6	383.5	35.3%
Notebook	215.7	169.1	-5.9%
Netbook	18.3	0.3	-65.4%

Figur 3. Canalys uppskattning på mobila apparaters försäljning år 2016 jämfört med siffrorna år 2012 (Canalys, 2012)

Detta betyder dock inte att man bara kan koncentrera sig på utveckling för enbart mobila apparater, utan behovet för webbsidor för normala datorer kommer att finnas kvar. En responsiv sida är inte heller alltid det bästa alternativet, utan ibland kan en ”äkta” mobil sida vara ett bättre alternativ. En sådan sida kan ha ett helt annat innehåll, design, och interaktivitet beroende på var användaren befinner sig, användarens apparat och anslutningshastighet och andra varierande element. Ett praktexempel kan vara en pizzeria med flera lokaler. Pizzerian kan ha en standardsida för stationära bordsdatorer och en skild mobil version som kan ha mobila egenskaper som baserar sig på användarens position och hjälper honom att hitta fram till närmaste pizzerian. En sådan lösning kräver mera än vad bara en responsiv sida kan erbjuda.

Men fastän alla webbsidor inte kräver en skild mobilsida med specifika egenskaper för olika apparater, kan det ändå vara bra att skräddarsy innehållet beroende på apparatens skärmstorlek då man bygger en responsiv sida. I apparater med små skärmar kan man ha det viktigaste innehållet överst och mindre viktiga delar längst nere på sidan eller gömma dem helt och hållet. Också storleken på menyerna kan vara bra att förstora för apparater med mindre skärmar för att underlätta navigationen när man använder fingrar istället för en noggrann mus. Allt detta är saker som man blir tvungen att testa och få

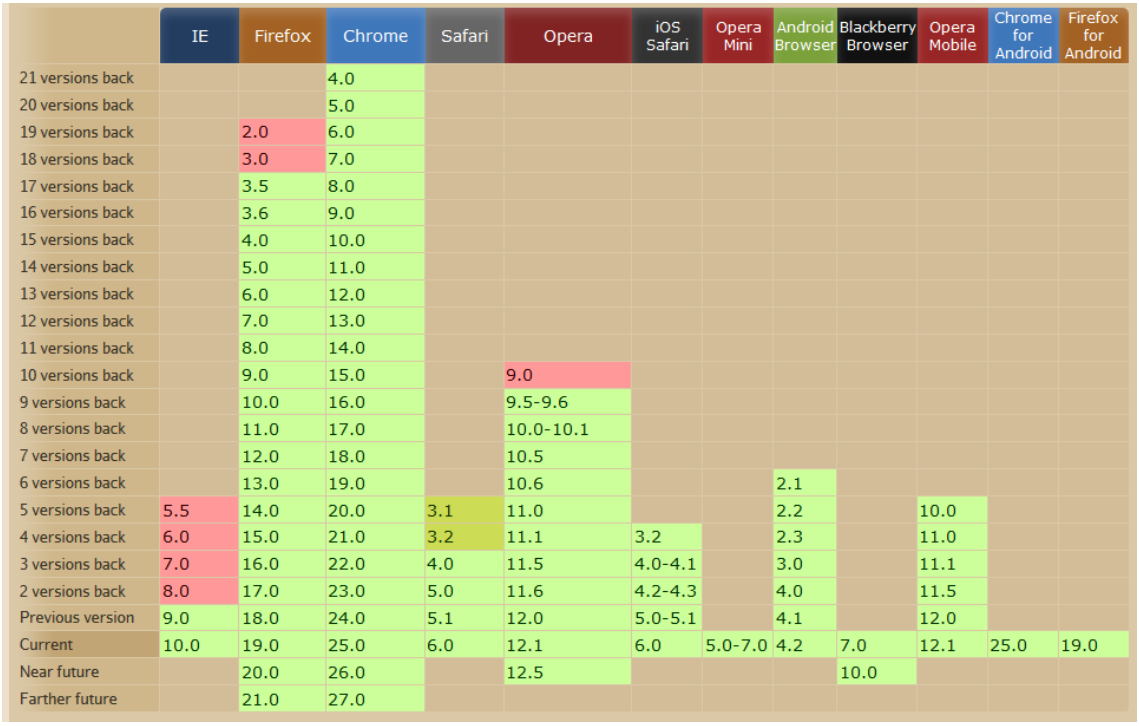
användarfeedback för att nå ett resultat som tillfredsställer användare med olika apparater. (Frain, 2012)

2.1 Responsiva designens element

Termen responsiv webbdesign gjordes känd av Ethan Marcotte i hans artikel ”Responsive Web Design” i webbtidningen ”A list apart” år 2010. Han satt ihop tre existerande element i en enhetlig uppfattning och kallade det för responsiv webbdesign. De tre elementen var *media queries, flexibel grid* och *flexibel bild.* För en fungerande och lyckad responsiv webbsida krävs alla dessa element, men Marcotte och andra utvecklare har hävdat att en äkta responsiv metod är mera än att bara ändra på layouten beroende på användarens skärmstorlek. Istället är det mera att ändra på hela tankesättet vid planering av webbsidor. Istället för att börja med en bestämd bredd för bordsdatorer och börja minska innehållet utgående från det, ska vi börja från de minsta och efter det utvidga sidan för större skärmstorlekar. (Frain, 2012, Marcotte, 2011)

2.1.1 Media queries

Media queries är en CSS3 teknik som tillåter oss att specificera en bestämd CSS stil beroende på vilka egenskaper skärmen på apparaten som besöker sidorna har. T.ex. med bara några rader kod kan vi ändra på hur sidan ser ut beroende på skärmstorleken, bildförhållandet, bildinriktingen m.m. I bilden nedan kan vi se att alla nyaste webbläsare stöder media queries, så det finns ingen orsak att inte använda dem.



	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
21 versions back			4.0									
20 versions back			5.0									
19 versions back		2.0	6.0									
18 versions back		3.0	7.0									
17 versions back		3.5	8.0									
16 versions back		3.6	9.0									
15 versions back		4.0	10.0									
14 versions back		5.0	11.0									
13 versions back		6.0	12.0									
12 versions back		7.0	13.0									
11 versions back		8.0	14.0									
10 versions back		9.0	15.0		9.0							
9 versions back		10.0	16.0		9.5-9.6							
8 versions back		11.0	17.0		10.0-10.1							
7 versions back		12.0	18.0		10.5							
6 versions back		13.0	19.0		10.6			2.1				
5 versions back	5.5	14.0	20.0	3.1	11.0			2.2		10.0		
4 versions back	6.0	15.0	21.0	3.2	11.1	3.2		2.3		11.0		
3 versions back	7.0	16.0	22.0	4.0	11.5	4.0-4.1		3.0		11.1		
2 versions back	8.0	17.0	23.0	5.0	11.6	4.2-4.3		4.0		11.5		
Previous version	9.0	18.0	24.0	5.1	12.0	5.0-5.1		4.1		12.0		
Current	10.0	19.0	25.0	6.0	12.1	6.0	5.0-7.0	4.2	7.0	12.1	25.0	19.0
Near future		20.0	26.0		12.5				10.0			
Farther future		21.0	27.0									

Figur 4. Browser stöd för media queries (Caniuse, 2013)

Media queries fungerar på ett väldigt enkelt sätt. I följande kodsnutt finns en enkel CSS snutt som ändrar färgen på sidans bakgrund beroende på skärmstorleken

```
body {
    background-color: black;
}

@media screen and (max width: 960px) {
    body {
        background-color: red;
    }
}

@media screen and (max width: 768px) {
    body {
        background-color: yellow;
    }
}
```

Figur 5. Ett enkelt exempel på en fungerande media query

I detta exempel skulle bakgrundsfärgen på sidan vara gul ända fram till 768 pixlar breda skärmar, röd till 960 pixlar breda och svart till ännu bredare skärmar. Dessa förändringar enligt skärmstorleken eller andra egenskaper kallas för ”breakpoints”. Vi kan också kombinera flera media queries genom att använda nyckelordet ”and” flera gånger t.ex.:

```
@media screen and (min-device-width: 480px) and (orientation: landscape) {···.}
```

Figur 6. Exempel på en media query som kombinerar flera egenskaper

I exemplet ovan skulle då apparater med över 480 pixlars bredd och en horisontell bildinriktning få de egenskaper som skulle definieras inom klammerparenteserna. Media queries fungerar på samma sätt som normala CSS-attribut d.v.s. den sista stilen man anger överskrider de tidigare stilar som påverkar samma element. T.ex. först gör man normala länkar för stora skärmar och senare skriver en stil med större länkar för mindre skärmar. Dock kommer alltid den noggrannaste angivna stilen att överskrida alla andra stilar, helt som i normala CSS fall. Följande media queries finns till förfogande:

- **width –** aktiva områdets bredd t.ex. bredden på webbläsaren
- **height –** aktiva områdets höjd t.ex. höjden på webbläsaren
- **device-width** – skärmbredden i själva apparaten
- **device-height** – skärmhöjden i själva apparaten
- **orientation –** bildinriktning, vertikal eller horisontell
- **aspect-ratio** – bildförhållandet i aktiva området t.ex. bildförhållandet i webbläsaren
- **device-aspect-ratio** – bildförhållandet i själva apparaten
- **color** – färgdjupet. Om apparaten inte har färger, är värdet 0.
- **monochrome** – färgdjupet i en monokrom apparat
- **resolution** – skärmupplösning d.v.s. hur många pixlar per tum apparaten har
- **scan** – bildåtergivning, d.v.s. hur bilden i apparaten skapas.
- **grid** – ger förmågan att veta om apparaten är gridbaserad eller bitmapbaserad

2.1.2 Flexibel grid layout

Tidigare byggde man oftast webbsidor på en fixed grid d.v.s. bredden på sidan var bestämd i förhand. Ännu för några år sedan när man visste pixeldimensionerna på de då populära smarttelefonerna och surfplattorna såsom iPad och iPhone, kunde man med media queries göra en bestämd layout för dem. Fastän detta ännu är möjligt och ett alternativ att alltid lägga till en ny layout till sidan beroende på vilken apparat som besöker sidorna och bara använda media queries, tillåter en flexibel grid layout att innehållet skalar sig relativt med den skärmstorleken som besökaren i sin webbläsare har. Detta sker så länge tills en media query säger att nu ska en ny layout implementeras. I princip betyder det att istället för att använda bestämda pixelstorlekar i våra CSS stilar, använder vi oss av relativa storlekar som t.ex. procent.

2.1.3 Flexibel bild och media

För en responsiv sida behöver vi också bilder och videon som skalar sig till den rätta storleken. Detta är tekniskt en enkel uppgift. För bilder behöver vi bara deklarera följande stil i vår CSS:

```
img {
    max-width: 100 %;
}
```

Figur 7. CSS-attribut för flexibla bilder

Detta gör att bilden är alltid 100 % av det element som bilden är innanför. Vidare kan man deklarera detta:

```
img, object, video, embed {
    max-width: 100%;
}
```

Figur 8. CSS-attribut för flexibel media

Då innehåller deklarationen också all annan media som sidorna möjligtvis innehåller. Detta är ett lätt sätt att göra bilder till rätt storlek, men det finns en annan sak man måste

tänka på. Fastän man skalar en bild, kommer den att ha samma filstorlek som den ursprungliga bilden, vilket kan orsaka längre laddningstider för mobila apparater i 3G nätverk. Man kan lösa detta med olika JavaScript eller CSS tekniker:

```
@media (min-width: 501px) and (max-width: 767px) {
   #image-container {
      max-width: 182px;
   }
   .image {
      background-image: url(image5.jpg);
   }
}
@media (min-width: 501px) and (max-width: 767px) and (-webkit-min-device-pixel-ratio:1.5) {
   .image {
      background-image: url(image7.jpg);
   }
}
@media (min-width: 501px) and (max-width: 767px) and (-webkit-min-device-pixel-ratio:2) {
   .image {
      background-image: url(image8.jpg);
   }
}
```

Figur 9. CSS-teknik för att välja en rätt storleks bild för rätt apparat.

CSS-metoden ovan fungerar så, att man i sin HTML-struktur skapar en div som fungerar som behållare för bilden. Inne i denna div skapar man en till div som har en bakgrundsbild som väljs med CSS beroende på apparatens skärmupplösning och denna bakgrundsbild kommer att fungera som den ”riktiga” bilden man vill visa.
–*webkit-min-device-pixel-ratio* är inte en officiell media query, men stöds av både iOS och Android och kan därför användas för att koncentrera en egenskap till dessa apparater. Exemplet ovan är inte fullständig och skulle bara påverka aparater mellan 501 och 767 pixlar och borde fortsättas vidare för att påverka apparater och webbläsare utanför dessa storlekar. Denna metod används bl.a. i Amazons produktsidor. (Smashing Magazine, 2013)

Det finns också färdiga verktyg som t.ex. Adaptive Images. Adaptive Images är ett program skapat av Matt Wilcox och finns på adaptive-images.com. Programmet skapar temporära bilder av olika storlekar av ursprungsbilderna och laddar upp en lämplig storleks bild åt användaren beroende på användarens skärmstorlek.

3 OPTIMERING AV WEBBSIDOR

För att bygga sidor till mobila apparater, är det bra att bekanta sig med de modernaste teknikerna som finns för att dra full nytta av en mobilapparats egenskaper. Utvecklingen sker hela tiden mot en teknik som underlättar skapandet av sidor för mobila apparater. T.ex. en smarttelefon har inte samma processorkraft och minne som en bordsdator och då kan stora sidor som baserar sig på mycket JavaScript bli för tunga och bete sig långsamt. Enligt HTTP Archive, som konstant studerar de 10 000 mest besökta webbsidor, är en sidas medelstorlek på webben 1,3 MB, vilket är 35 % större än förra året och när man surfar med en mobilapparat och räknar med mobiloptimerade sidor är medelstorleken 720 kB. Då en användare besöker sju sidor per dag, kommer den totala dataöverföringen då redan att ligga på 150 MB i månaden, vilket kan överskrida en dataöverföringsgräns i vissa abonnemang. För att ge en snabbare och trevligare användarupplevelse för mobila användare, och p.g.a. av i detta projekt inte kommer att bygga en skild mobiloptimerad sida, kommer jag att gå igenom några tekniker som man kan använda när man bygger en responsiv sida. Enligt en annan undersökning av Keynote Systems, som är ett företag som koncentrerar sig på lösningar att mäta, testa och förbättra prestanda av olika innehåll på webben, är 80 % av människorna som surfar på nätet med en mobilapparat besvikna över upplevelsen och skulle använde sina smarttelefoner mera om användarupplevelsen skulle förbättras. Då också 64 % av användarna vill att sidorna ska laddas under 4 sekunder, till skillnad av en 9 sekunders medelladdningstid för tillfället, är dessa tekniker bra att känna till. (The Star, 2013, Keynote, 2012)

Annat som webbutvecklare också sällan vill hamna att göra är webbläsar- och multiplattform -testning d.v.s. testa hur olika lösningar ser ut på olika webbläsare och operativsystem. För att undgå detta bör man från början bekanta sig med vilka tekniker är stödda och använda ramverk som t.ex. jQuery mobile, som stöder en mängd olika mobila plattformer. (Hadlock, 2012)

3.1 HTML5 och CSS3

HTML5 är märkspråket för att strukturera och presentera innehåll i webben och en viktig teknologi i Internet. HTML5 är den femte versionen av själva HTML standarden

som skapades år 1990. Många av de nya egenskaperna har blivit byggda för att kunna köras på mindre effektiva apparater, så som smarttelefoner och surfplattor. Nya element som <video> och <canvas> har en viktig roll då man bygger webbsidor som ska fungera på mobila apparater. Dessa två element tillåter webbutvecklare att bygga sidor med funktioner som förr krävde flash, något som några typer av dagens mobila apparater inte stöder. HTML5 har också nya egenskaper som offline stöd d.v.s. man kan spara information på webbläsarens cache (temporära minne), för att ge en användare möjlighet att jobba på en sida med en dålig eller ingen anslutning. Andra HTML5 egenskaper som mobilwebbutvecklare kan dra nytta av är geolokalsation som ger användaren möjlighet att berätta för webbläsaren var han befinner sig för en mera lokaliserad användarupplevelse och web worker som är JavaScript skript som kan jobba i bakgrunden av sidan i flerkärniga apparater och göra användningen av en sida snabbare

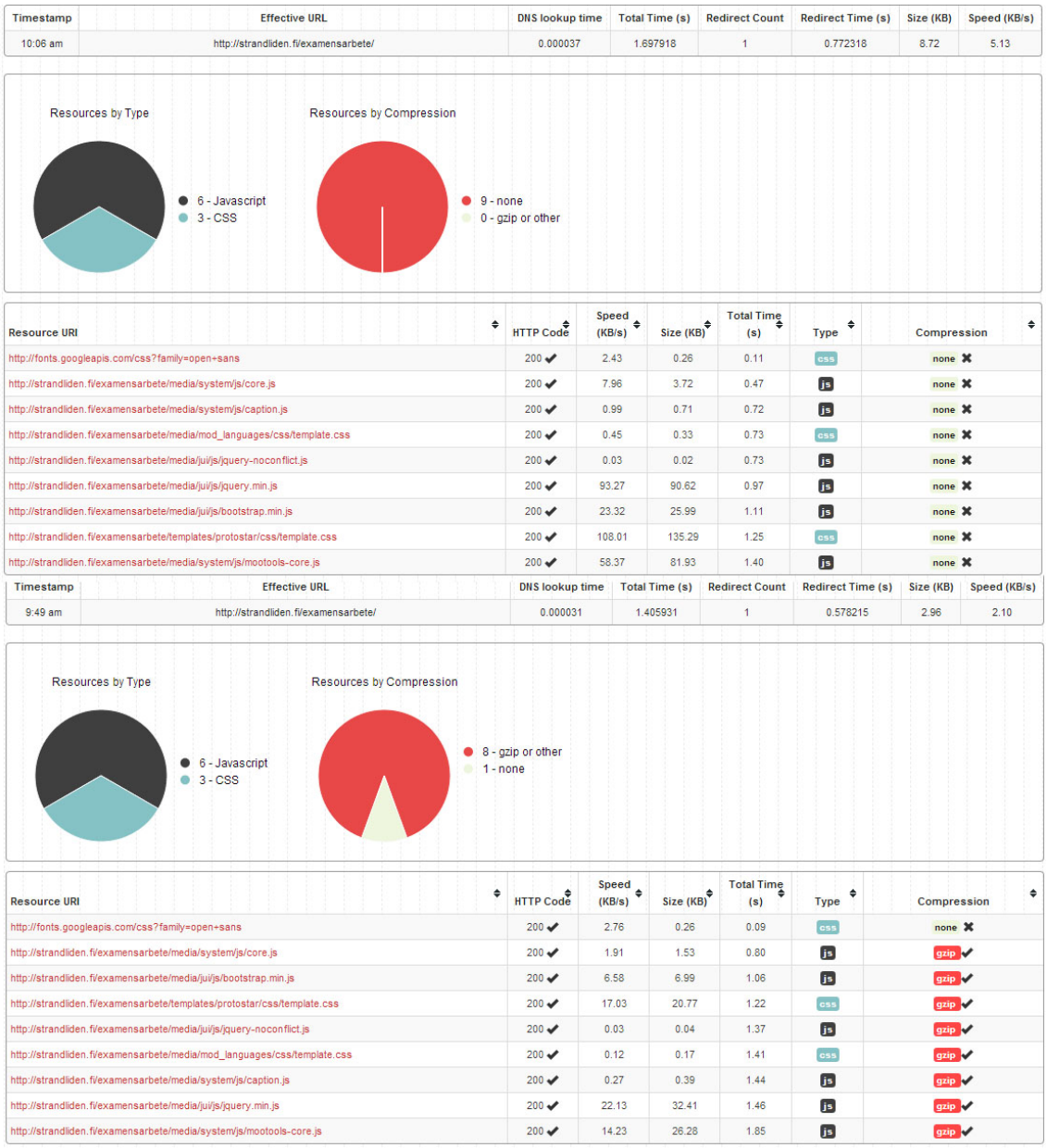
HTML5 är inte för tillfället den officiella standarden för HTML, men alla de största webbläsarna stöder HTML5, så det finns ingen orsak att inte använda HTML5. I dagens läge finns det en tendens att sätta in all ny webbteknik innanför samma ramar och kalla det för HTML5. Som exempel ser man ofta SVG (Scalable Vector Graphics), en vektorgrafik-format, att bli nämnas som en medlem i HTML5 teknikerna, fastän det är en över tio år gammal självständig grafikspecifikation.(Wikpedia, 2013a, Lawson & Sharp, 2012)

CSS3 är den senaste versionen av stilmallsspråket CSS, som innehåller stilar för att bestämma hur HTML-element presenteras. CSS3 ger egenskaper som förut krävde användning av bilder. Runda kanter, animationer, skuggor och andra grafiska effekter blir mycket snabbare att visas upp då man använder sig av de nya teknikerna som stöds av alla moderna webbläsare.

3.2 HTTP komprimering

HTTP komprimering fungerar med samma princip som när man komprimerar en normal fil. I webben sker komprimeringen på serversidan och webbläsaren expanderar filen. Allt detta händer väldigt snabbt och är alltid ett bättre sätt att sända information. En stor del av webbhotellen gör det redan färdigt. På sidan gzipwtf.com kan man granska om

filerna komprimeras eller inte och efter en granskning på adressen strandliden.fi/examensarbete där jag har mitt projekt påbörjat, ser jag att webbhotellet jag använder inte har komprimeringen färdigt inställt. Då är det lättaste sättet att säkra att filerna komprimeras, att kopiera några rader från HTML5 Boilerplates *.htaccess* –fil som man hittar från HTML5 Boilerplates hemsida och bakom *source code* länken. HTML5 Boilerplate är ett projekt som koncentrerar sig på att hjälpa webbutvecklare att bygga bättre webbsidor och applikationer. Koden som man ska kopiera finns under *Compression* delen i deras *.htaccess* –fil. Koden innehåller inställningarna till många olika servrar och kan användas som sådan, eller om man vet vilken server webbhotellet använder sig av, kan man lämna bort de inställningar som påverkar andra servar. I nästa figur ser vi skillnaden som sidan gzipwtf.com ger, utan och med HTTP komprimering för mitt projekt i det tillstånd som det för stunden är:



Figur 10. HTTP komprimeringsresultat (gzipwtf.com, 2013)

I den övre mätningen är komprimeringen inte på och den totala tiden det tog att söka sidan var 1,7 sekunder, medan komprimeringen är på i den lägre mätningen och söktiden var 1,4 sekunder. Här blev skillnaden bara 0,3 sekunder, men ju större sidorna är, desto större blir skillnaden. HTML5 Boilerplate har också andra egenskaper definierade i deras *.htaccess* –fil som kan öka hastigheten på nedladdningen av sidor, som t.ex. sparandet av filer i webbläsararens temporära minne. (Wikipedia, 2013b, CSS-Tricks, 2013)

3.3 Icke-förstörande komprimering av bilder

Man kan minska bilder på många sätt, men oftast resulterar en förminskning också i att man tappar information, d.v.s. kvalitén försämras. Med en icke-förstörande komprimering menar man att bilderna minskas utan att någon information försvinner. Det finns program som klarar av en icke-förstörande komprimering och fastän bilderna jag i detta projekt använder inte kommer att vara stora, kommer jag att köra alla bilder genom ett gratisprogram som heter PNGgauntlet, som fungerar på Windows. Till Mac OS finns ett liknande gratisprogram som heter ImageOptim. (CSS-tricks, 2013)

3.4 Kombinerande av CSS- och JavaScript –filer

Orsaken varför vissa sidor tar längre att ladda på mobila apparater är inte själva nedladdningshastigheten på internetanslutningen. Enligt en undersökning av PCWorld tidskriften är medelnedladdningshastigheten för 3G nätverk i Europa ca 2 Mbps vilket betyder ca 0,24 MB/s, som inte då är flaskhalsen. 3G nätverk klarar bra av att visa videon från t.ex. YouTube på ens smarttelefon, men till synes enkla webbsidor kan ta en lång tid att uppvisas. Orsaken är den höga latensen som de mobila bredbanden har. Varje objekt eller fil som browsern begär efter, kan ha en latens på 190ms. D.v.s. en YouTube video kräver bara en begäran av servern och då blir latensen irrelevant, medan en webbsida kan bestå av hundratals filer vilket kan leda till flera sekunders sökningstid. Ett lätt sätt att minska på antalet filer webbläsaren blir tvungen att söka, är att kombinera sina CSS- och JavaScript filer. Man kan också minska på antalet bilder webbläsaren blir tvungen att söka genom att kombinera bilderna i en fil och med hjälp av CSS söka rätta området på bilden som webbläsaren sedan visar för slutanvändaren. (PCWorld, 2012, Web Performance Today, 2012)

Det finns många fler sätt att optimera en sida för en snabbare användning i mobilapparater, men att gå genom alla tekniker skulle kräva ett skilt examensarbete. De ovannämnda teknikerna är lätta att använda och ger redan goda resultat i sig.

4 VERKTYG OCH TEKNIK

Jag kommer att utveckla en responsiv sida med Joomla. Joomla är ett innehållshanteringssystem som uppdaterades till Joomla 3.0 i september 2012. Den nya versionen är byggd mera för mobila apparater och innehåller ett front-end ramverk som heter Twitter Bootstrap. Jag kommer att behandla de ovannämnda noggrannare i de följande kapitlen, men för att effektivt kunna utveckla en webbsida behövs vissa webbutvecklingsverktyg. Som källkodsediterare kommer jag att använda Sublime Text 2. Den har vissa egenskaper som underlättar editerandet, t.ex. en ”Livereload” plugin som uppdaterar de ändringar man gör i källkoden direkt i webbläsaren d.v.s. man behöver inte manuellt uppdatera webbsidan på webbläsaren för att se vilka ändringar i sidans utseende som skett. Som webbläsare kommer jag att använda Chrome p.g.a. att jag är van att använda de färdigt inbyggda ”Developer Tools” (utvecklingsverktyg) som Chrome har i sig. Utvecklingsverktygen är små verktyg med vilka man kan undersöka koden i en webbsida. Om jag högerklickar på en webbsida i Chrome och trycker på ”Granska komponent”, kan jag undersöka hela den kod som bygger upp det innehåll som jag ser framför mig. Detta underlättar hittandet av den rätta lilla kodsnutten från Bootstraps över 6000 rader långa CSS fil, som påverkar ett litet område på sidans utseende.

4.1 Twitter Bootstrap

Twitter Bootstrap är en samling på HTML, CSS och JavaScript tekniker som underlättar utvecklingen av webbsidor. Bootstrap är också det front-end ramverk som finns färdigt implementerat i Joomla 3. Bootstrap utvecklads av Mark Otto och Jacob Thornton från Twitter för att hjälpa att sammankoppla förekommande Internetverktyg. Före Bootstrap fanns det för många verktyg och bibliotek och när webbutvecklare behövde hjälp och tips för sina problem, fanns det hundratals olika lösningar runtom i webben vilket lätt ledde till kaotiska inkonsekventa resultat. Detta var problemet som Bootsrap utvecklarna Otto och Thornton ville lösa. I augusti 2011 publicerades Bootstrap och gjorde direkt ett gott inflytande bland webbutvecklare. Detta är mycket förståeligt då Bootsrap är ett noggrant utvecklat ramverk som innehåller de viktigaste elementen i webbutvecklingen men ändå är lätt att modifiera. I januari 2012 publicerades Twitter Bootstrap 2.0 med stora förbättringar speciellt för responsiv layout och i dagens läge är det mest visade

projektet på Github med över 33 000 visningar, över dubbelt mera än den näst mest sedda. Detta är ingen liten merit med tanke på andra stora projekt som HTML5 Boilerplate och jQuery som också finns på Github. (Cochran, 2012)

4.1.1 Vad Bootstrap innehåller

För tillfället (1.4.2013) är Bootstrap i version 2.3.1. Bootstraps hemsidor på twitter.github.com/bootstrap bjuder på exempelkod, källkod och möjlighet för att utvidga och skräddarsy Bootstrap för egna behov. Själva installationen av Bootstrap innehåller tre mappar: en *css*, en *img* och en *js* -mapp. *css* mappen innehåller fyra CSS stilmallar:

- *bootstrap.css* – basstilmallen för Bootstrap
- *bootstrap.min.css* – en minimerad version av basstilmallen
- *bootstrap-responsive.css* – stilmallarna med responsiva egenskaper
- *bootstrap-responsive.css* – en minimerad version av responsiva stilmallen

img mappen innehåller två polygonbilder som innehåller små ikoner som kan användas i Bootstrap. *js* mappen innehåller två JavaScript filer, en bas JavaScript fil med allt som behövs och en minimerad version. Den minimerade versionerna är tänkt för den slutliga webbsidan då de har mindre filstorlek och därmed snabbare att ladda ner för slutanvändaren. Själva utvecklingen med minimerade versionerna är besvärlig då alla texter, attribut, funktioner m.m. är ihoppressade och svårlästa. Bootstraps JavaScript är beroende av jQuery, så om man vill använda Bootstraps JavaScript, måste man lägga till en länk till jQuery biblioteket i sin sida.

4.2 Joomla!

Joomla, eller Joomla! med ett utropstecken, är ett innehållshanteringssystem baserat på öppen källkod och som de flesta open source projekt är det under fortsatt utveckling. Det är gjort för människor för att publicera innehåll i Internet: nyheter, bloggar, bilder, produkter, dokument, evenemang eller hundratals annat innehåll. Det har varit i framgångsrikt bruk i över sju år och är ett populärt CMS bland miljontals användare. Ordet

Joomla är härledd från ordet Jumla på Swahili vilket betyder ”alla tillsammans”. Ursprungligen var Joomla skapad av australienare, men i dagens läge utveckals Joomla av frivilliga programmerare runtom i hela världen. Joomla används av stora företag som, McDonald’s, Sony, General Electric, eBay, Palm, Ikea, Kellogg’s, Porsche och flera andra. (Burge, 2012)

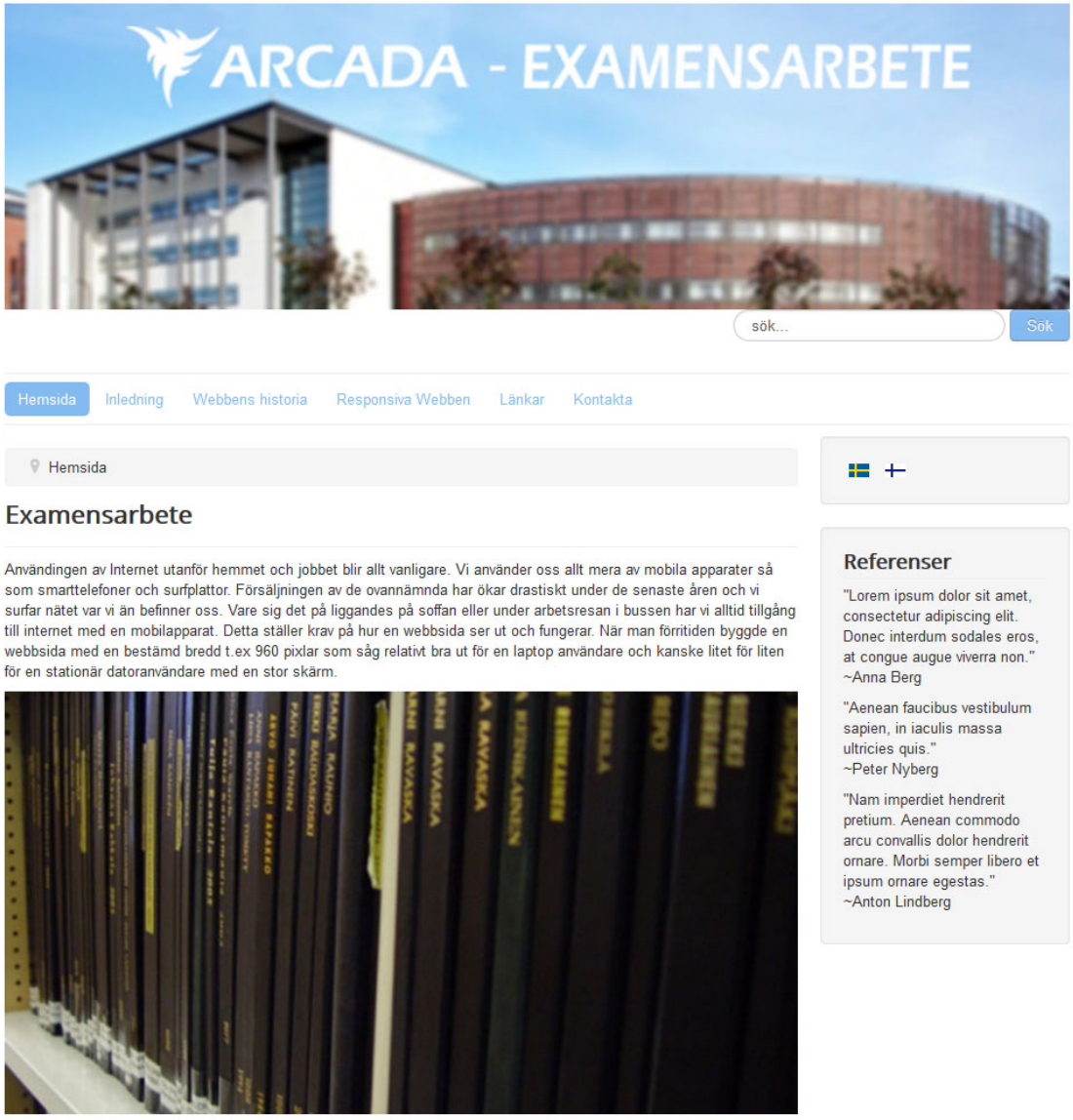
Figur 11. Franska företaget Danones hemsidor på danone.com är byggda med Joomla. Sidorna är dock inte responsiva och kommer säkert snart att förnyas. (Danone 2013)

Joomla 3 publicerades 29 september 2012 och i den nya versionen kommer Twitter Bootstrap färdigt implementerat, vilket underlättar skapandet av responsiva sidor med Joomla. Detta betyder att inte bara själva webbsidan som syns för slutanvändaren är responsiv, utan också administrativa sidan är responsiv och lätt användbar på en mobil apparat vilket underlättar underhållandet av webbsidan. Joomla kan köras lokalt på egen dator, då krävs att man har en server, databas och PHP skriptspråket installerat på datorn, eller så kan Joomla köras på en webbserver. I detta arbete kommer Joomla att köras lokalt på en Windows dator och senare att överföras till ett webbhotell för allmän åtkomst.

5 WEBBUTVECKLING I JOOMLA!

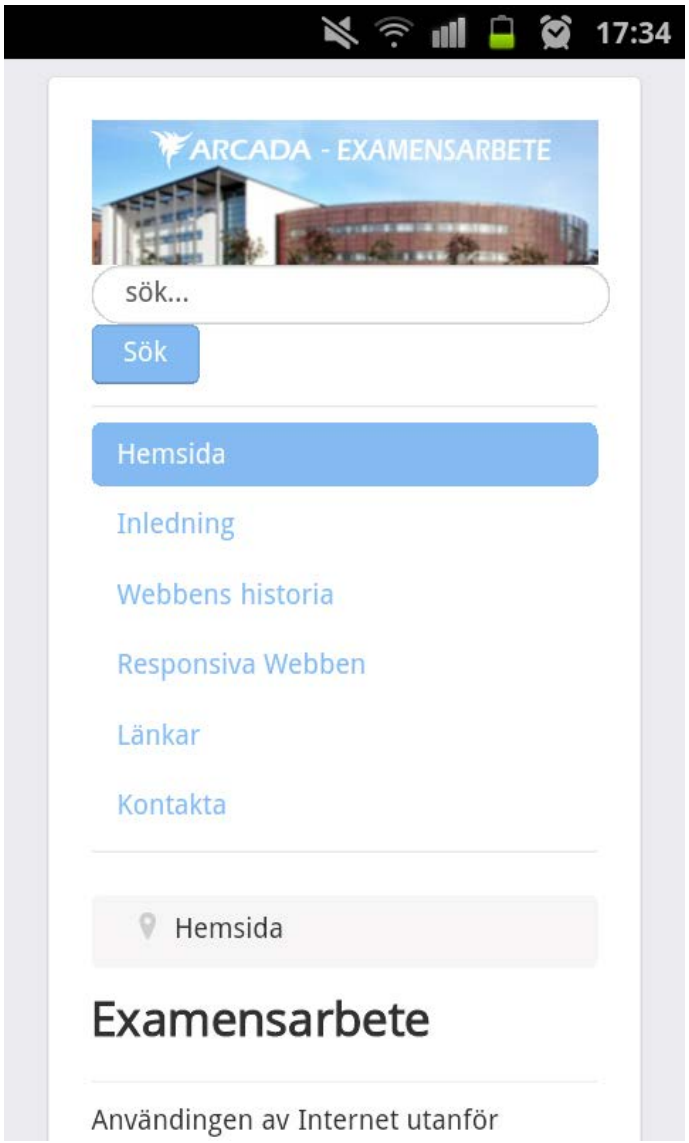
5.1 Joomla, Bootstrap och Protostar

När man installerar Joomla 3 kommer den färdigt implementerat med Bootstrap. En botten för en sida i Joomla kallas ”template” och den bastemplate som Joomla 3 kommer med heter Protostar. Protostar i sig grundar på Bootstrap, vilket betyder att den är redan färdigt responsiv, dock måste man göra stora förändringar i den för att få en sida att se ut och bete sig så som man själv vill. Jag har byggt en sida som grund för min responsiva utveckling. Som jag redan nämnde i avgränsningen, kommer jag inte att gå igenom hur man lägger till artiklar, bilder eller moduler till ens sida från Joomlas kontrollpanel. Webbsidan nedan är byggd genom att enbart använda Joomlas kontrollpanel och inga CSS- eller template -filer har blivit ändrade.



Figur 12. En enkel sida byggd på Joomlas bastemplate, Protostar.

På webbsidan ovan finns en stor header-bild, en sökfunktion, navigation med rull-gardinsmenyer, en språkmodul, en modul på sidans högra sida med text. I footern ligger inloggningsmöjligheten och i mitten av sidan finns en artikel med en bild. I figuren ser vi hur sidan beter sig på en normal persondator med hög skärmupplösning. I nästa bild ser vi hur sidan ser ut på en smarttelefon med en skärmupplösning på 480 x 800 pixlar.



Figur 13. Samma sida som på figur 6 på en smarttelefon

Vi ser att fastän Protostar -templaten redan i sig fungerar responsivt utan några konfigureringar, finns det behov för vissa förändringar. Men för att få en sida att se ut och bete sig som man själv vill måste man bygga en egen template. T.ex. ett av den här sidans största problem är, att rullgardinsmenyn inte fungerar på en smarttelefon. Detta innebär att, som sidan nu är byggd, vi inte har någon åtkomst till sidorna som ligger under menyerna ”Inledning”, ”Webbens Historia” och ”Responsiva Webben”, eftersom de alla fungerar som rullgardinsmenyer. Men, som vi ser, fungerar Joomla responsivt direkt man tar den i bruk och använder bastemplaten Protostar. Protostar använder Bootsraps

responsiva stilegenskaper och genom att undersöka källkoden i Protostar hittar man snabbt en bra grund för en egen responsiv design i Joomla.

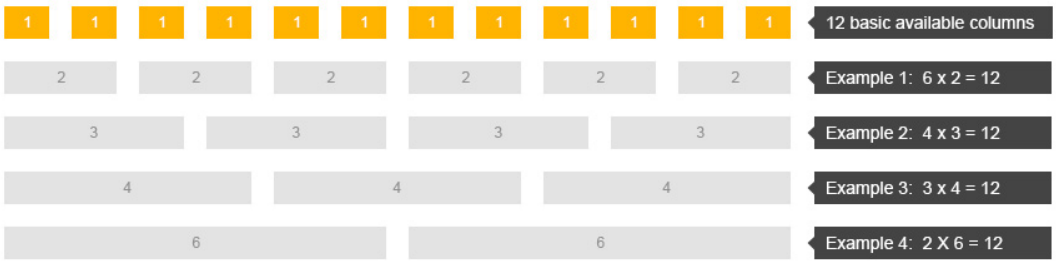
5.2 Bootstraps gridsystem

För att kunna utveckla sidor i Joomla som baserar sig på Bootstrap, är man tvungen att förstå hur Bootstraps gridsystem fungerar i samarbete med Joomla. Bootstraps basgrid är en 940 pixlar bred fixed grid utan några responsiva egenskaper. Man kan också använda den som en anpassningsbar grid när man använder den responsiva CSS – stilmallen med gridden. Då anpassar sig gridden enligt användarens skärmstorlek. Bootstrap erbjuder också en fullt responsiv flexibel grid som heter ”fluid grid”. I Joomlas kontrollpanel kan man välja vilkendera grid man vill använda ifall man gör det möjligt i ens *index.php* -fil. *Index.php* -filen behandlas närmare i följande kapitel. Bootstrap använder ett 12-kolumns gridsystem. Det finns ett klassattribut som heter *span* som ökar från *span1* till *span12* beroende på hur många kolumner av gridden vi vill ge till ett block. *span12* -klassen är då en klass med bredden på hela webbsidan, *span6* är hälften och *span3* är en fjärdedel osv. Om vi exempelvis vill skapa en ny div och ge den hälften av sidans utrymme skriver vi följande:

```
<div class="row">
    <div class="span6">...</div>
    <div class="span6">...</div>
</div>
```

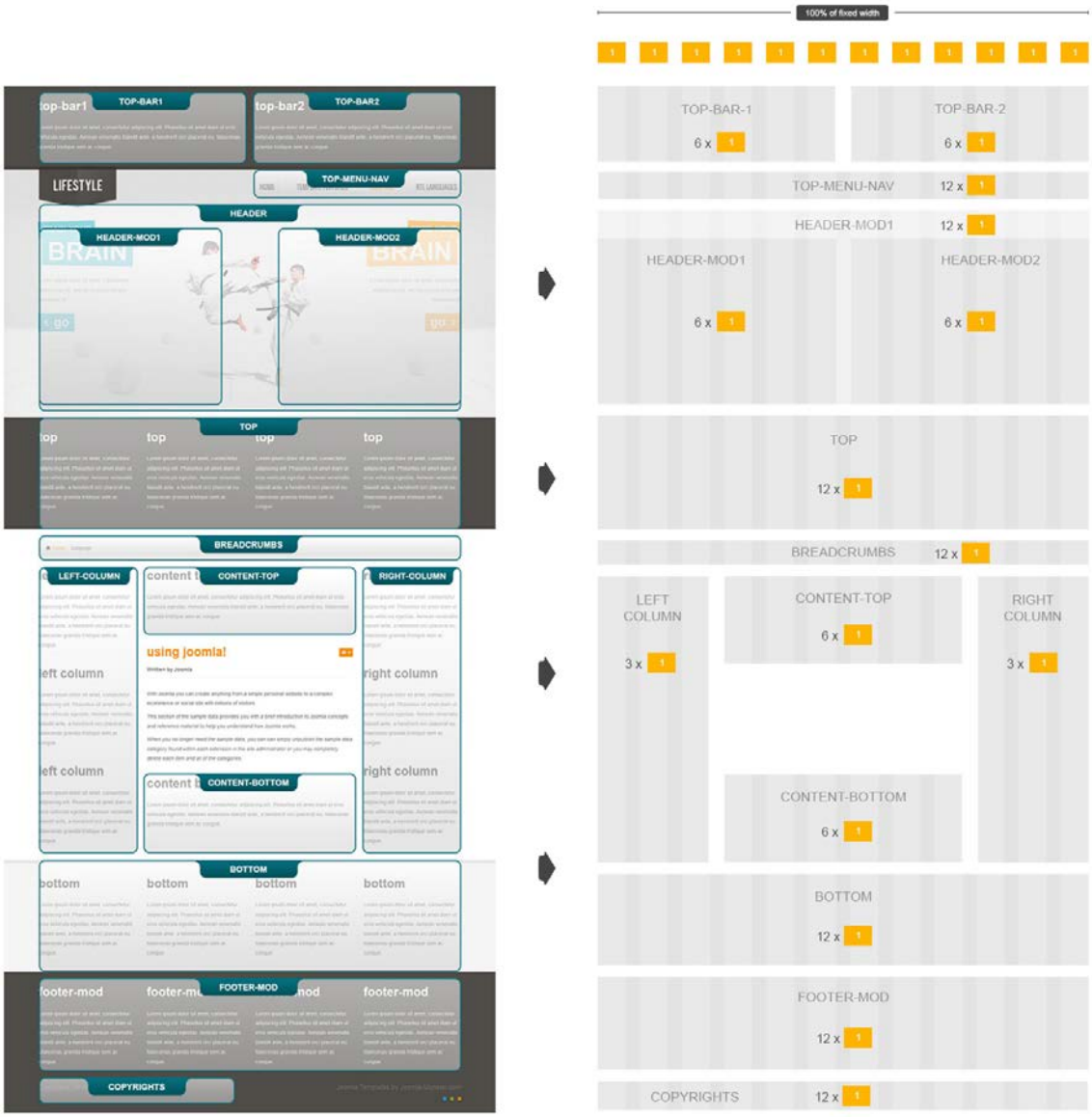
Figur 14. Exempel på hur man skapar en ny rad med två lika stora kolumner i Bootstrap

Först skapar man en div av klassen *row* som betecknar att det kommer en ny rad på sidan och därefter skapar man en ny kolumn på raden och säger hur mycket utrymme man vill att kolumnen ska få. Då man bara använder row klassen är det fråga om en fixed grid. I följande figur har vi ett exempel på ett färdigt grid-system i Bootstrap som använder en fixed grid.



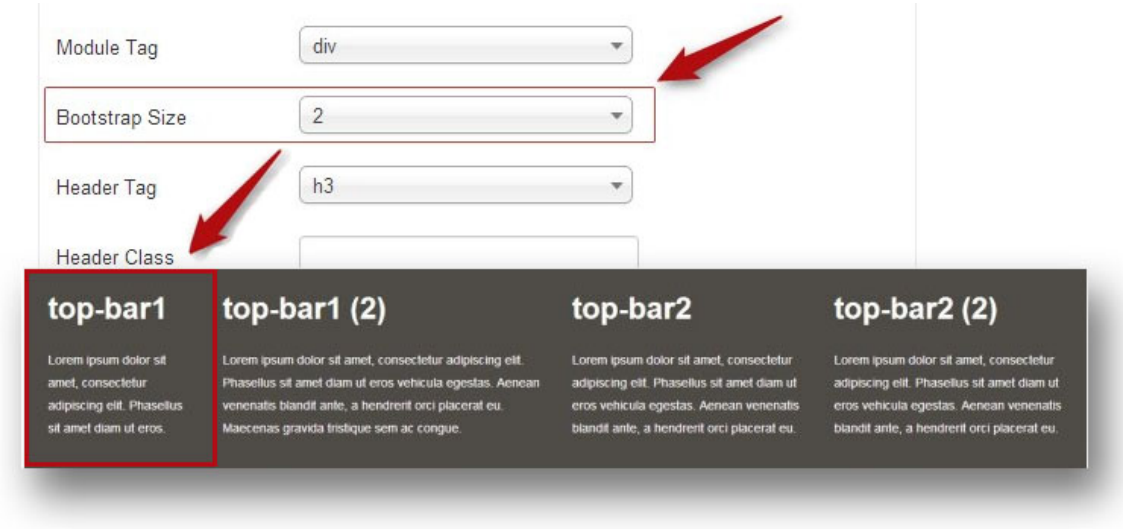
Figur 15. Twitter Bootstraps grid-system med jämna kolumner. Alla rader måste innehålla 12 kolumner (Joomla Monster, 2013)

I figuren ovan kan vi på första raden se 12 stycken divar där varje div är av klassen *span1*. På följande rad har vi divar av klassen *span2* vilka det ryms sex stycken av i Bootstraps 12-grid-system. Varje Bootstrap -rad i en fixed grid måste innehålla 12 kolumner. En exempelsida där vi använder Bootstraps fixed grid i Joomla kunde se ut såhär:



Figur 16. Ett exempel på hur en webbsida kan delas med Bootstraps fixed 12-kolumnssystem (Joomla Monster, 2013)

I Joomla använder man moduler för att skapa innehåll till sidorna. Så istället för 12 kolumner kan man tala om 12 modulpositioner. Alla rader i detta fixed grid -system skapas med olika kombinationer av 12 modulpositioner. Detta innebär också, att när vi tar in moduler från Joomlas kontrollpanel, måste vi i modulens inställningar ändra Bootstraps size -alternativ till sådana storlekar att modulernas gemensamma storlek inte överskrider 12. Om man inte vill ge användaren möjlighet att ändra på modulens storlek, kan man hårdkoda den i *index.php* –filen



Figur 17. Modulplacering från Joomlas kontrollpanel (Joomla Monster, 2013)

I figuren kan vi se en header som är 12 kolumner bred. Inne i headern finns fyra moduler. I figuren skapas första modulen och om man vill ha en två kolumner bred modul, väljer man Bootstrap Size: 2 i kontrollpanelen om man lämnat det möjligt. Resten av modulerna är av storleken fyra, tre och tre, då blir den totala bredden 12 kolumner eller modulpositioner. I en flexibel fluid grid fungerar detta på ett annat sätt. Vi delar hela sidan i procentuella delar som tillsammans är 100 %. Hela sidan innehåller igen 12 baskolumner, men varje del som vi delar sidan i innehåller också 12 kolumner. I praktiken betyder det att om vi delar webbsidan i två delar som båda är 50 % av sidbredden kommer båda delarna tillsammans att innehålla 24 modulplaceringar. I nästa figur har vi samma layout som tidigare, men i en fluid grid.

Figur 18. Exempel på en Joomlasida med Bootsraps fluid grid som bas (Joomla Monster, 2013)

I figuren ser vi tydligast skillnaden i mittersta delen av sidan där left- och right -kolumn båda använder 25 % av bredden, och content- top och bottom tar 50 % av bredden, men alla innehåller 12 modulplaceringar. För att skapa en fluid grid i Bootsrap används klassen *row-fluid*. Om vi ännu demonstrerar samma header som vi har i figur 9, kommer det att se ut på följande sätt:

Figur 19. Samma headerlayout som i figur 9, men i en flexibel grid (Joomla Monster, 2013)

I exemplet ovan är en fluid-grid delad i två 50 % delar. Då innehåller båda delarna 12 modulpositioner och för att få samma utseende på första modulen som i fixed grid -exemplet, väljer vi Bootstrap size 4 från Joomla kontrollpanelen istället. När skärmstorleken blir mindre kommer utseendet att förändras. Raderna beter sig responsivt enligt de egenskaper man angett i Bootstraps CSS -filer. I fixed grid kollapsar raderna i standardinställningen på skärmar som är under 767 px breda och i fluid grid kollapsar de responsivt enligt skärmens storlek på det sätt som syns i följande figur. Kollapsandet sker dock endast om man har med de responsiva egenskaperna som finns i *responsive.css* stilmallen. (Cochran, 2012)

Figur 20. Bootstraps gridsystem kollapsar i mindre skärmostorlekar

5.3 Skapandet och innehållet av template -mappen

För att kunna bygga en design för en responsiv sida eller vilken sida som helst i Joomla, är man tvungen att bygga en egen template. Man kan också editera en färdig template som någon annan har byggt och jobba vidare på den, men i mitt fall kommer jag att bygga en template från noll. Jag skapar den i en komprimerad arkivfil, d.v.s. en zip -fil som sedan installeras i min Joomlainstallation via Joomlas kontrollpanel. Arkivfilen kan ha varierande innehåll, i mitt fall kommer följande mappar och filer inkluderas: -en *css* -mapp med min egen CSS -fil som jag gett namnet *custom.css*, en *images* -mapp med bilder, en favicon som är den lilla ikonen som befinner sig bredvid adressfältet, *template_preview.jpg* och *template_thumbnail.jpg* -bilderna som Joomla använder för att förhandsvisa hur templaten ser ut, och *templateDetails.xml* -filen som innehåller en lista på allt som min template -mapp kommer att innehålla, en beskrivning på webbsidan, t.ex. namnet på sajten och vem som skapat templaten och en lista på alla moduler som kommer att finnas på sidan. I templateDetails -filen kan man också skriva ut andra egenskaper som man vill att sajten ska innehålla, men tillsvidare klarar vi oss med detta.

Template -mappen innehåller också en *index.php* -fil som är grunden för hela webbsidan som kommer att byggas. Härnäst kommer jag att visa hur min *index.php* ser ut tills-

vidare, och förklara vad de olika kodsnuttarna betyder. Till först har jag följande PHP kod:

```php
<?php
defined('_JEXEC') or die;
JHtml::_('bootstrap.framework');
JHtmlBootstrap::loadCss($includeMaincss = true);
?>
```

Figur 21. Laddning av Bootstrap ramverket och Bootsrap CSS filerna i index.php filen

Första raden är en kontroll på att det är frågan om en Joomla –fil, andra raden laddar Bootstraps JavaScript -ramverk som kommer att användas i sidorna, och tredje raden laddar alla de Bootstrap CSS-filer som nämndes i kapitlet om Bootstrap plus några tilläggs CSS-filer som är gjorda för Joomla. Vi kan ladda Bootstrap-filerna direkt på vår dator eller server, men med hjälp av detta uttryck försäkrar vi oss i att vi använder oss av den nyaste Bootstrapversionen. Efter det kommer HTML- dokumenttypsdeklarationen och HTML -headelementet som ser ut som följande:

```html
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<jdoc:include type="head" />
    <!--[if lt IE 9]>
<script src="<?php echo $this->baseurl ?>/media/jui/js/html5.js"> </script>
    <![endif]-->
<link href='http://fonts.googleapis.com/css?family=Ubuntu' rel='stylesheet' type='text/css'>
<link href="css/custom.css" rel="stylesheet" type="text/css">
</head>
```

Figur 22. Head elementet på index.php filen

Jag kommer att använda HTML5 som standard för HTML–märkspråket, i det har dokumenttypsdeklarationen förenklats till det som finns på första raden. Femte raden med viewport är en viktig rad för responsiva sidor. Den berättar att sidan är optimerad för mobila apparater och hur innehållet ska skalas på skärmen. Bootstraps responsiva funktioner kräver också att man har denna metatag med i källkoden. Efter det laddar jag dynamiskt head –elementet, som innehåller t.ex. sidans namn, från Joomla. If-satsen efter det är för Internet Explorer version 8 och tidigare. Satsen ger dessa versioner en Java-

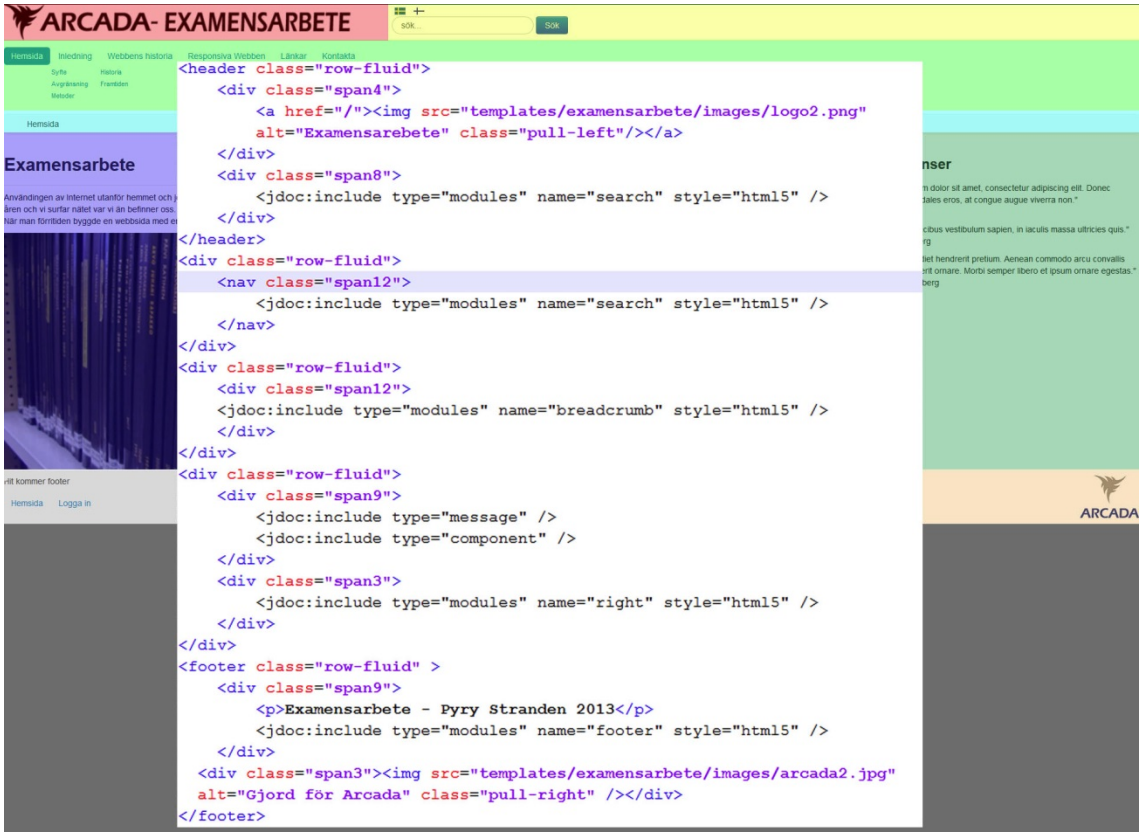
Script -fil som hjälper dem att förstå HTML5 -standarden. Följande kodsnutt laddar en Google Font, Ubuntu, som jag senare kan använda. Till sist refererar jag till min egen CSS-fil som jag kommer att använda för att modifiera utseendet och beteendet på min sida. Filen måste laddas sist så att de stilar jag definierar i denna fil, överskrider de som definierats i Bootstraps egna CSS. Efter head -elementet kommer själva HTML-innehållet. Detta kommer att bestå av en Bootstrap flexible grid. Som exempel kan jag visa hur header -taggen ser ut:

```
<header class="row-fluid">
    <div class="span4">
        <a href="index.php" > <img src= "tem-plates/examensarbete/images/logo2png"
        alt="Examensarbete" class="pull-left" /> </a>
    </div>
    <div class="span8">
        <jdoc:include type="modules" name="search" style="html5" />
    </div>
</header>
```

Figur 23. Header elementet på index.php filen

På detta sätt bygger man en responsiv sida med en flexibel grid som redan tidigare beskrivits i Bootstrap gridsystem -kapitlet. Jag har en header av klassen *row-fluid* vilket innebär att den drar ut sig enligt skärmens bredd och innehåller två divar: en med bredden fyra kolumner, *span4*, och en med bredden åtta kolumner, *span8*, som då tillsammans fyller upp Bootstraps 12 -kolumners grid. Vi kunde också med ett PHP skript göra det möjligt att välja i Joomlas kontrollpanel om man vill använda en fixed grid eller fluid grid. I vårt fall vet jag att jag kommer att jobba med en fluid grid så jag skriver det färdigt här. I första diven har vi en logo på vänstra sidan vilken jag åstadkom med Bootstrap –klassen *pull-left* och i andra diven har vi en modul. *Pull left* är klassen man ger ett element för att placera det på vänstra sidan i det innehållande elementet. Namnet på modulerna spelar ingen roll, men jag använder namn som beskriver modulernas funktion. Med hjälp av *style=html5* -deklarationen kan vi i Joomlas kontrollpanel använda de nya HTML5 -namnen i divarna, så som *nav*, *aside*, *article* osv. Detta är inte ett måste, men är en trevlig ny egenskap som Joomla 3 har. Efter header -elementet kommer resten av HTML-sidan, navigationen, själva innehållet d.v.s. artikeln, högra spalten på webbsidan och footern. Allt detta byggs enligt samma koncept som headern, genom att skapa en flexibel div som innehåller de olika modulerna. När jag skapat allt detta har

jag en färdig botten för en Joomla template som jag sedan komprimerar till en ZIP-fil och installerar via Joomlas kontrollpanel. Här följer en figur som visar hur sidan ser ut när templaten är installerad och innehållet i *index.php* -filen som styr innehållet:

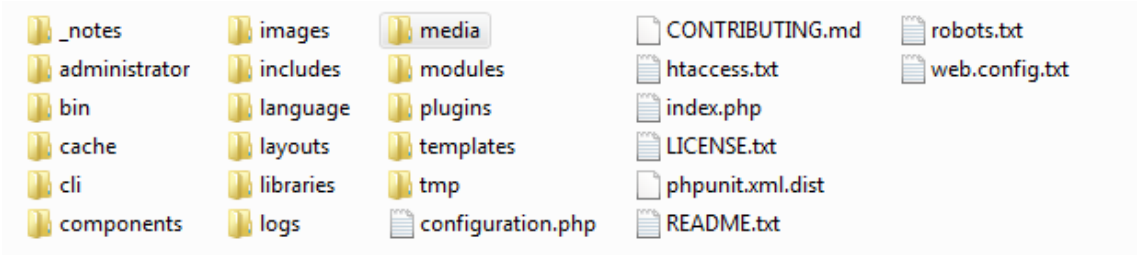


Figur 24. index.php filens innehåll och sidans utseende

Som man ser har innehållet dragit ut sig på hela sidans bredd och de olika färgerna representerar de olika divarna. Vi kan tydligt se hur Bootstraps flexibla grid fungerar och hur lätt det är att placera de moduler man vill ha på ungefär sina rätta platser. Till näst är det bara att ändra på vår egen CSS-fil och se hur vi ändrar på CSS stilen som Bootstrap här färdigt skapat till vår förfogande. *Index.php* filen kommer att befinna sig i template foldern och man kan senare ge mera egenskaper t.ex. placera mera modulpositioner genom att ändra på filen.

5.4 Stilisering av de olika elementen med CSS

Som jag redan visade i min *index.php* -fil, kommer jag att ladda ner Bootstraps CSS -filer som en bas för sidorna. Joomla innehåller alla de fyra CSS -filer som tillhör Bootstrap och ännu två tilläggs- CSS -filer. Dessa filer heter *bootstrap-extended.css,* som behövs för att överskrida vissa färdiga egenskaper som en Jooomla -sida har, och *bootstrap-rtl.css,* som är en förkortning av ”right-to-left” och är en stilmall för språk som skrivs från höger till vänster och kommer inte att behövas i vårt fall. För att bättre förstå vad vi ska ändra på för att få vår sida att se ut som vi vill, bör vi ta en titt på Joomlas mappstruktur som ser ut som följande:



Figur 25. Joomlas mappstruktur

Största delen av mapparna är självförklarande. Bootstrap -filerna befinner sig i *media* -mappen. Vi kan göra våra ändringar direkt inom den, men alltid när Joomla uppdaterar sig kommer alla ändringar vi gjorde där att nollställas. Därför kommer vi att dra nytta av *cascading* möjligheten i CSS-stilen. När vi tar ett titt på *index.php*, ser vi till att vår egna *custom.css* -fil laddas efter Bootstraps CSS –filer. Då kommer alla förändringar som vi gjort i *custom.css* -filen att överskrida de inställningar som vi har i Bootstraps CSS-filer. *Custom.css* -filen befinner sig i templates -mappen där alla de andra filerna som vi i förra kapitlet skapade också befinner sig. Det är här vi kommer att arbeta för att skräddarsy vår sida för våra behov. *Configuration.php* används för att konfigurera databaskopplingen och andra inställningar som är viktiga för sidans funktion.

5.4.1 Navigationen

Bootstrap erbjuder färdigt konfigurerade rullgardinsmenyer. Från Twitter Bootstraps sidor kan jag direkt kopiera den kod som finns under *Responsive navbar*, eftersom det är en responsiv meny som intresserar oss, och klistra den i min *index.php*. Den enda

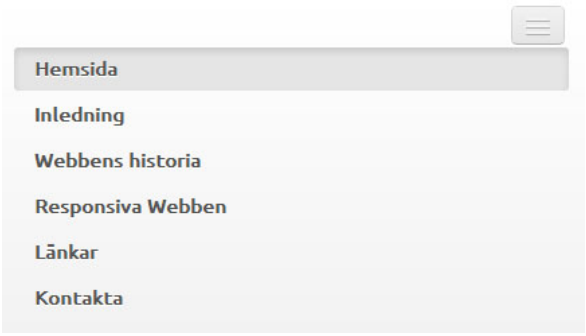
ändringen som jag gör är att byta namnet på första elementet från *div* till *nav* eftersom jag vill använda HTML5 -elementnamn. Koden ser ut som följande:

```
<nav class="span12 navbar">
    <div class="navbar-inner">
        <div class="container-fluid">
            <a class="btn btn-navbar" data-toggle="collapse" data-target=".nav-collapse">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            </a>
            <div class="nav-collapse collapse">
                <jdoc:include type="modules" name="navigation" style="none" />
            </div>
        </div>
    </div>
</nav>
```

Figur 26. Källkoden för navigationen

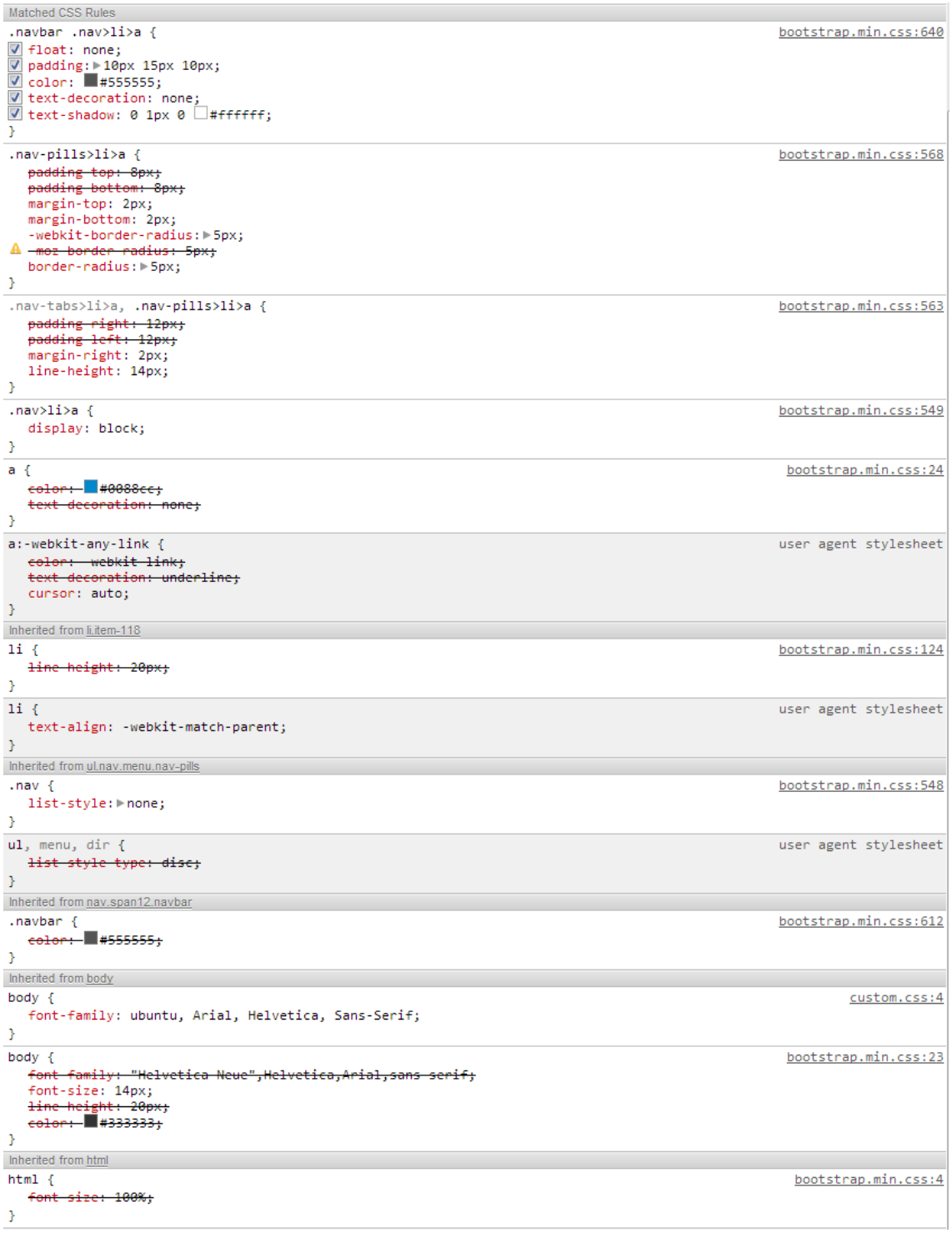
Som vi ser har vi ett *nav*-element som är av klassen *span12*, d.v.s. det kommer att använda hela sidans bredd. Vi använder fluid-klassen för det element som innehåller navigationen för att få det att fungera responsivt. Efter det använder vi oss av Bootstraps inbyggda JavaScript. Vi behöver bara lägga till *data-toggle=collapse* och *data-target* till ett element, i vårt fall en knapp som heter *btn-navbar*, för att ge elementet kontroll över ett nedfällbart element. *Data-target* -attributet tar emot en CSS-stil och ger den vidare till det elementet som vi har specificerat, i vårt fall till det element som innehåller navigationsmodulen och är av klassen *nav-collapse*. Efter det lägger vi klassen *collapse* till vårt element som innehåller sidans navigation. Resultatet med några stilförändringar är en responsiv meny som ser ut som följande i de olika skärmupplösningarna: (Twitter Bootstrap, 2013)

Figur 27. Navigationen på en persondator med hög skärmupplösning



Figur 28. Navigationen i en mindre skärmupplösning och menyn i det nedfällda tillståndet

För att ytterligare ändra på utseendet av navigationen måste vi gräva djupare i Bootstraps stilmallen. De går lättast genom att använda Chromes utvecklingsverktyg. När jag högerklickar på min navigation och väljer ”granska komponent” i Chrome, kan jag undersöka allt som påverkar hur navigationen ser ut och beter sig. Chromes utvecklingsverktyg visar den HTML-kod som skapar sidan och de CSS -egenskaper som påverkar utseendet. Jag är mest intresserad av vilka CSS egenskaper jag måste ändra på och var de befinner sig. Chrome ger följande resultat för navigationen:



Figur 29. CSS-attributen som Chromes utvecklingsverktyg visar

Detta underlättar avsevärt arbetet i stora stilmallar som kan vara svåra att läsa och förstå. På vänstra sidan ser jag namnet på den egenskap som jag vill undersöka eller möjligen ändra på och på högra sidan ser jag i vilken fil egenskapen befinner sig. Som vi ser är det en lång lista egenskaper och största delen av dem befinner sig i *bootstrap.min.css*

-stilmallen. Det är dock inte i den filen jag ska göra mina ändringar, eftersom, som jag tidigare skrev, skulle dessa ändringar nollas då Boostrap kommer ut med en uppdatering som Joomla automatiskt laddar. De ändringar som jag vill göra ska utföras i min egna *custom.css* –fil, den kommer att överskrida alla de egenskaper som finns i Bootstraps stilmall. Ett exempel där vi kan se en stil som överskrider en Boostrap stil är *font-family* –egenskapen som jag ändrat på i min egna stilmall till *ubuntu, Arial, Helvetica, Sans-Serif* och den överskrider standard- fontfamiljen som Bootstrap har definierat. Om jag t.ex. vill att mina länkar i navigationen ska vara i versaler, behöver jag enbart lägga till följande egenskap i min *custom.css* -stilmall:

```
.navbar .nav> li> a {
    text-transform: uppercase;
}
```

Figur 30. Ändring av navigationen till versaler

Jag kommer att göra små andra ändringar i navigationens utseende, men de är inte relevanta för navigationens responsiva beteende.

5.4.2 Huvudinnehållet

För att stilisera vårt innehåll måste vi igen ta en titt med Chromes utvecklingsverktyg vad det är som styr utseendet på vår sida. Som tidigare nämnts, baserar sig vår sida på Bootstraps flexibla gridsystem. Vi kan dock inte börja ändra på Bootstraps stilmall som styr gridden, t.ex. genom att ändra på *span* -klassen, för det kan söndra all responsivitet som vi har i sidorna, eller orsaka andra oönskade förändringar på griddens struktur. Vi bör ändra på stilen på de element som Joomla skapar när man placerar innehåll i sidorna. Dessa element hittar vi igen med utvecklingsverktyget.

5.5 Stilmöjligheter för smarttelefoner och surfplattor

Hittills har jag inte använt mig av en enda media query. All responsivitet som finns på sidorna bestäms av de media queries som är definierade i Bootstraps stilmallar. Ett problem som finns på sajten är navigationen. Den fungerar ännu också som en rullgardinsmeny, vilket inte kommer att fungera så bra på en smarttelefon eller surfplatta.

När jag öppnar utvecklingsverktygen i Chrome och minskar på webbläsarens storlek, visar utvecklingsverktyget när en break point i media queryn sker, och vilken stilmall då kommer i bruk. I Bootstrap sker detta vid 979 pixlars bredd. Vi har en media query som säger:

```
@media (max-width:979px)
```

Efter denna rad kommer alla de stilförändringar som påverkar skärmar med bredden mindre än 979 pixlar. Rullgardinseffekten kommer med följande CSS-attribut:

```
.nav-child {
    display: none;
}
.nav-child li:hover {
display: block;
}
```

Figur 31. CSS för navigationen

Navigationen är ett element på en oordnad lista (ul) av klassen *nav-child*. Med *display none* -attributet kommer listan inte att visas. När man för musen över en länk i navigationen aktiveras *li:hover* –klassen, och menyn under länken visas med *display:block* -attributet. För att få listan med länkarna att vara synlig hela tiden behöver jag bara lägga till en media query med följande innehåll:

```
@media (max-width:979px) {
.nav-child {
display: block;
    }
}
```

Figur 32. CSS för att göra navigationen synlig i skärmar under 979 pixlar

Nu syns navigationen oberoende om musen är på en länk eller inte. Några andra stilförändringar bör också göras, t.ex. förstoring av fontstorleken för att underlätta träffandet av de länkar som man trycker på med fingrarna, samt för en allmänt behagligare användarupplevelse. Vi kan emulera en apparat med mindre skärmupplösning genom att

minska på webbläsarens fönster. Detta fungerar dock endast om media queryn är definierat med enbart *width*. Om den istället är definierad med *device-width* kommer skärmens egentliga skärmupplösning att tas i beaktande och webbläsararens storlek har ingen betydelse. Detta kan också lösas med Chromes utvecklingsverktyg i *settings -> overrides -> device metrics*. Fastän detta visuellt kan ge en uppfattning om hur navigationen ser ut, bör noggrannare tester med olika mobila apparater utföras.

När jag kör tester med en smarttelefon märker jag, att fastän navigationen ser bra ut, framstår ett annat problem. Logon som jag skapat i Photoshop beter sig nog responsivt till en viss mån, men på en smarttelefon är den ändå för stor och söndrar hela sidans konstruktion. Det här beror på att, som tidigare nämnts, Bootstraps grid kollapsar och detta orsakar att logon blir för stor för en rad. Bootstrap har färdigt implementerat några klasser i sin stilmall för att visa eller dölja innehåll beroende på vilken apparat man besöker sidorna med. Egentligen är dessa inte beroende av apparaten, utan igen av skärmupplösningen och kan överskridas i en skild stilmall. I standarkonfigurationen fungerar klasserna på följande sätt:

Tabell 1. Bootstraps responsiva nyttoklasser (Twitter Bootstrap, 2013)

Klass	Telefon 767 px och under	Surfplatta 979 px till 768 px	Persondator standard
.visible-phone	synlig	dold	dold
.visible-tablet	dold	synlig	dold
.visible-desktop	dold	dold	dold
.hidden-phone	dold	synlig	synlig
.hidden-tablet	synlig	dold	synlig
.hidden-desktop	synlig	synlig	dold

För att få en skild logo att synas för mobila apparater kan jag dra nytta av det här. Jag behöver endast lägga till klassen *hidden-phone* till min nuvarande logo i *index.php* -filen. Därefter kan jag göra en mindre logo och lägga den i *index.php* -filen med klassen *visible-phone*. Efter det lägger jag till följande media query:

```
@media (max-width: 767px) {
    header {
        text-align: center;
    }
}
```

Figur 33. Centrera logo för mindre skärmstorlekar

Då har jag en centrerad logo som bara syns i telefoner. Det som också kunde ändras i en mobil vy är att ta bort all onödig marginal från sidornas kanter. Denna marginal underlättar läsandet på en datorskärm och är visuellt attraktiv, men på en liten skärm tar den onödigt utrymme. För tillfället är templaten också konfigurerad så att oberoende om det finns innehåll i en modul eller ej, kommer webbläsararen att lämna så mycket utrymme, som man i templaten ursprungligen har definierat. Som exempel finns det en modul som heter *right* som är tre kolumner bred, men inte alltid har innehåll på varje sida. P.g.a. hur denna rad är konstruerad, d.v.s. en rad med en modul av klassen *span9* och en av *span3* kan man inte heller bara skriva en if-sats som frågar, om modulen finns, placera den på sidan, men om den inte finns, hoppa över den. Om modulen inte skulle finnas, skulle de resultera i en endast nio kolumner bred modul och tre tomma kolumner. Istället måste man skriva en PHP if-sats som ändrar på den första modulens innehållande elementets bredd om den andra modulen inte finns och en if-sats som döljer modulens innehållande element om det inte finns någon modul. Slutliga delen som innehåller huvudinnehållet i sidan, kommer att se ut på följande sätt:

```
<div class="row-fluid">
    <?php if ($this->countModules('right')): ?>
    <div class="span9">
        <?php else : ?>
        <div class="span12">
        <?php endif; ?>
            <jdoc:include type="message" />
            <jdoc:include type="component" />
        </div>
        <?php if($this->countModules('right')) : ?>
        <div class="span3">
            <jdoc:include type="modules" name="right" style="html5" />
        </div>
        <?php endif; ?>
    </div>
</div>
```

Figur 34. PHP koden för att dölja tomma moduler

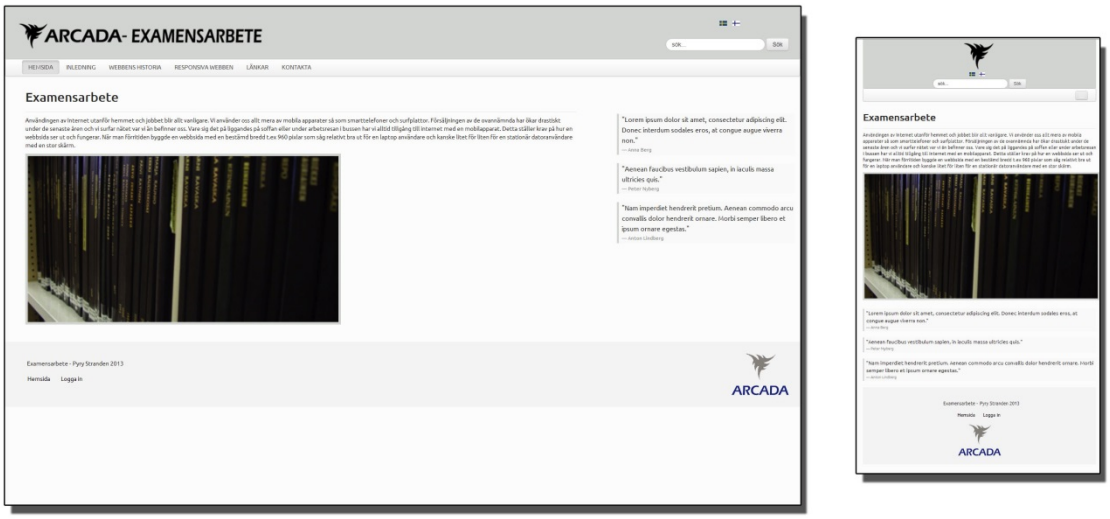
Det finns också andra små förändringar, t.ex. att centrera den lilla navigation som finns på footern. Den kan inte centreras med ett enkelt *text-align center* -attribut utan måst centreras genom att ge navigationen den rätta bredden för att den ska vara i mitten av footern. Alla dessa visuella förändringar är lätta att göra då man använder sig av Sublime Text *livereload* funktion som hela tiden visar hur de förändringar man gör påverkar sidans utseende. Med små förändringar kan jag slutföra designen på själva sidan.

5.6 Administration och upprätthåll med mobila apparater

Joomlas administratorsida använder sig på samma sätt som framsidan en responsiv template direkt efter installationen. Templaten heter Isis. På samma sätt kan man också bygga en egen template för administratorsidan som jag gjorde till framsidan. Då jag loggar in mig på administrator med en mobilaparat konstaterade jag att den inte fungerar helt som förväntat. På surfplattan fungerade inte övre navigationen, men med navigationen på högra sidan kunde jag komma framåt till de önskade områdena och göra samma ändringar som på en bordsdator. Med smarttelefonen fungerade övre navigationen som en kollapsande navigation som den ska, men själva innehållet på vissa sidor var bristfälligt. För att editera en artikel i Joomla, ska man först trycka på en *checkbox* för att välja artikel och sedan trycka på editera. Checkboxen var inte synlig i smarttele-

fonen och ingen editering av artikeln kunde göras. På samma sätt som på framsidan kan man med Chromes utvecklingsverktyg hitta källkoden på vad som orsakar detta och hitta en *hidden-phone* klass som då gömmer denna checkbox från användare med en mindre än 767 pixlar bred skärm, men var denna klass befinner sig i Joomlas administratorfiler kunde jag inte hitta och modifiera. I själva skrivandet av en artikel betedde sig inte verktyget för skrivandet responsivt vilket orsakade att man blir tvungen att zooma in och ut och flytta på sidan i telefonen. Annars fungerar administratorsidan på ett sätt som går att använda, fastän det inte är helt smärtfritt på en smarttelefon. För äldre versioner av Joomla finns det program för mobiladministration som användare i Joomla gemenskapet gjort, men dessa fungerar inte ännu i Joomla 3.

6 RESULTAT OCH TANKAR



Figur 35. Slutliga sidan på en hög upplösning och på en låg upplösning

I detta slutarbete har jag studerat de viktigaste elementen i responsiv webbutveckling samt optimering av sidor för mobilapparater och använt dessa tekniker för att bygga en responsiv webbplats i Joomla innehållshanteringssystem genom att använda ett av de populäraste front-end ramverken, Bootstrap. Resultatet kan man se i figuren ovan och i adressen http://www.strandliden.fi/examensarbete. Själva designen kunde bearbetas vidare, men efter att ha planerat en sida från nolläge såhär långt, är det lätt att göra de

önskade ändringar med samma tekniker som använts tidigare. På en bred skärm kunde man dela sidan i flera spalter vilket är lätt att göra i Bootstrap. Joomla bjuder på mycket mera än vad jag i detta arbete kan gå igenom t.ex. flera utvidgningar som kan göra en responsiv sida mera interaktiv och innehållsrik. Man kan också ge användare möjligheten att ändra på saker i templaten genom att lämna egenskaper dynamiskt valbara för användaren. Dessa egenskaper kan vara färgen på headern, logon, typsnittet, eller andra små egenskaper, som man vill ha möjlighet att byta utan att kontakta en webbutvecklare. Men allt detta är utanför hur man bygger en responsiv sida och studerades därför inte närmare.

Jag har inte tidigare gjort sidor för smarttelefoner, men kan redan efter detta projekt konstatera att en smal skärm minskar designmöjligheterna ganska radikalt. Header, navigation, huvudinnehåll, sekundärt innehåll och footer är ganska långt designmönstret man blir tvungen att följa. Jag testade sidorna med de populäraste webbläsarna, Firefox, Chrome och Internet Explorer, med två smarttelefoner, HTC Desire Z och Samsung Galaxy S och en surfplatta Asus TFT300t. Jag fann inga problem med någon av dessa webbläsare eller apparat och sidorna fungerade konsekvent på alla. Detta är ingen överraskning eftersom Bootstrap i sina CSS- filer har färdigt webbläsarspecifika egenskaper för att allt ska fungera på rätt sätt. När man överskrider dessa egenskaper i sin egen stilmall behöver man bara komma ihåg att ta med alla de attribut som finns i Bootstraps stilmall, då fungerar allt som det ska. Laddningstiden på sidorna med mobilapparater är ganska lång, då optimeringen inte är lika lätt på en sida baserad på Joomla, som den skulle vara om man bygger en sida själv från noll. Jag använde alla de metoder som jag gick igenom i kapitel tre om optimering, men resultatet blev inte så bra som jag velat. Fastän man försöker minska på antalet begäran som webbläsaren gör till servern, har Joomla många filer som webbläsaren blir tvungen att söka, vilket förlänger sidans laddningstid. Att optimera CMS-baserade webbsidor för mobilapparater kunde vara ett lämpligt arbete att forska vidare i.

I mitt slutarbete ville jag också få fram om byggandet av en responsiv sida med Joomla binder designen en ”Joomlakännetecknande” design, d.v.s. att man direkt genom att se på sidorna kunna säga att sidan är byggd på Joomla. Jag har inte tidigare byggt en template till Joomla, men jag har sett flera templates som inte följer den basmall som man

kan känna igen en Joomla sida på. Då man bygger en template till Joomla med Bootstrap som grunden, ger den så många möjligheter man lätt kan ändra på, att det inte är något problem att få en individuell design. Dock om man använder en färdig navigation från Bootstraps exempelsida, kan den se likadan ut som andra sidor, men också i navigationens stilmall finns det otaliga möjligheter att ge den ett personligt utseende, bara man bekantar sig med alla de CSS3 möjligheter som medföljer. Det är inte heller Joomla som begränsar designen, för jag kunde använda samma design som jag gjorde i *index.php* – filen till en annan sida där innehållet inte skulle laddas dynamiskt i Joomla och stilisera det på samma sätt gjorts här. Sist och slutligen är Joomla bara ett innehållshanteringssystem som hjälper en användare att publicera innehåll, medan webbutvecklaren ansvarar för hur innehållet kommer att visas på sidan. Genom att förstå Bootstraps gridsystem och med en aning kunskap i CSS och JavaScript, finns det inga begränsningar på hurdana sidor man kan bygga på en Joomla baserad sida.

KÄLLOR

Burge Stephen, Joomla! Explained, Your Step-by-Step Guide, Addison-Wesley Professional, 2012
(ISBN: 978-0-321-70378-1)

Cochran David, Twitter Bootstrap Web Development How-To, Packt Publishing, 2012
(ISBN:1849518823)

Frain Ben, Responsive Web Design with HTML5 and CSS3, Packt Publishing, 2012
(ISBN: 978-9350237885)

Hadlock Kris, jQuery Mobile Develop and Design, Peachpit Press, 2012
(ISBN: 978-0-321-82041-9)

Lawson Bruce, Remy Sharp, Introducing HTML5, New Riders, 2012
(ISBN: 978-0-321-78442-1)

Marcotte Ethan, Responsive Web Design, A Book Apart, 2011
(ISBN: 978-0984442577)

Ruvalcaba Zak, Anne Boehm, Murach's HTML5 & CSS3, Mike Murach & Associates, 2012
(ISBN: 978-1890774660)

Canalys, Mobile device market reach 2.6 billion units by 2016 | 2013, [www], Hämtat 12.3.2013
http://www.canalys.com/newsroom/mobile-device-market-reach-26-billion-units-2016

Canalys, Smart phones overtake client PCs in 2011 | 2012, [www], Hämtat 12.3.2013
http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011

Caniuse, Can I use CSS3 media queries | 2013, [www], Hämtat 10.3.2013
http://caniuse.com/css-mediaqueries

CSS-Tricks, Chris Coyier, #114: Let’s Do Simple Stuff to Make Our Websites Faster | 2013, [www], Hämtat 4.42013
http://css-tricks.com/video-screencasts/114-lets-do-simple-stuff-to-make-our-websites-faster/

Global mobile statistics 2012 Part B: Mobile Web; mobile broadband penetration; 3G/4G subscribers and networks | 2012, [www], Hämtat 20.3.2013
http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/b

Google Trends | 2013, [www], Hämtat 5.3.2013
http://www.google.com/trends/explore#q=responsive web

ITWire, Stuart Corner, Mobile networks bringing broadband to developing countries | 2012, [www], Hämtat 12.3.2013
http://www.itwire.com/it-industry-news/market/57001-mobile-networks-bringing-broadband-to-developing-countries

Joomla | 2013, [www], Hämtat 5.3.2013
http://www.joomla.org

Joomla! Official Documentaion, What's new in Joomla! 3.0 | 2013, [www], Hämtat 12.3.2013
http://docs.joomla.org/What%27s_new_in_Joomla!_3.0

Joomla Monster, How to understand bootstrap grid system? | 2013, [www], Hämtat 10.3.2013
http://www.joomla-monster.com/knowledge-base-area/templates-documentation/general-info-framework-for-joomla-3-0/how-to-understand-bootstrap-grid-system

Keynote, 2012 Mobile User Survey | 2012, [www], Hämtat 4.4.2013
http://www.keynote.com/docs/reports/Keynote-2012-Mobile-User-Survey.pdf

Marcotte Ethan, Responsive Web Design, A List Apart | 2010, [www], Hämtat 13.3.2013
http://alistapart.com/article/responsive-web-design

PCWorld, Mark Sullivan, 3G/4G Performance Map: Data Speeds for AT&T, Sprint, T-Mobile, and Verizon | 2012, [www], Hämtat 4.4.2013
http://www.pcworld.com/article/254888/3g_4g_performance_map_data_speeds_for_atandt_sprint_t_mobile_and_verizon.html

Read and Write, Github Has Surpassed Sourceforge and Google Code 2011, [www], Hämtat 11.3.2013
http://readwrite.com/2011/06/02/github-has-passed-sourceforge

Smashing Magazine, Johan Johansson How To Make Your Websites Faster On Mobile Devices | 2013, [www], Hämtat 4.4.2013
http://mobile.smashingmagazine.com/2013/04/03/build-fast-loading-mobile-website/

Statscounter | 2013, [www], Hämtat 20.3.2013
http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-201002-201302

The Star, Oliviera Michael, Big web pages are bad news for mobile users | 2013, [www], Hämtat 6.4.2013
http://www.thestar.com/life/technology/2013/03/07/big_web_pages_are_bad_news_for_mobile_users.html

Twitter Bootstrap, JavaScript in Bootstrap Collapse | 2013, [www], Hämtat 24.3.2013
http://twitter.github.com/bootstrap/javascript.html#collapse

Twitter Bootstrap, Components Dropdown menus | 2013, [www], Hämtat 22.3.2013
http://twitter.github.com/bootstrap/components.html#dropdowns

Twitter Bootstrap, Responsive utility classes | 2013, [www], Hämtat 24.3.2013
http://twitter.github.com/bootstrap/scaffolding.html#responsive

Web Performance Today, Joshua Bixby, Mobile versus desktop latency | 2012, [www], Hämtat 4.4.2013
http://www.webperformancetoday.com/2012/04/02/mobile-versus-desktop-latency/

W3tech, Usage of content management systems for websites | 2013, [www], Hämtat 13.3.2013
http://w3techs.com/technologies/overview/content_management/all

Wikipedia, HTML5 | 2013a, [www], Hämtat 2.4.2013
http://en.wikipedia.org/wiki/HTML5

Wikipedia, HTTP Compression, 2013b, [www], Hämtat 4.4.2013
http://en.wikipedia.org/wiki/HTTP_compression